BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 December 2002 (12.12.2002)

PCT

(10) International Publication Number
WO 02/098292 A1

(51) International Patent Classification[7]: A61B 5/055

(21) International Application Number: PCT/AU02/00731

(22) International Filing Date: 6 June 2002 (06.06.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PR 5543 7 June 2001 (07.06.2001) AU

(71) Applicants *(for all designated States except US)*: **THE UNIVERSITY OF SYDNEY** [AU/AU]; Parramatta Road, Sydney, New South Wales 2006 (AU). **HOWARD FLOREY INSTITUTE OF EXPERIMENTAL PHYSIOLOGY AND MEDICINE** [AU/AU]; University of Melbourne, Parkville, Victoria 3052 (AU).

(72) Inventors; and

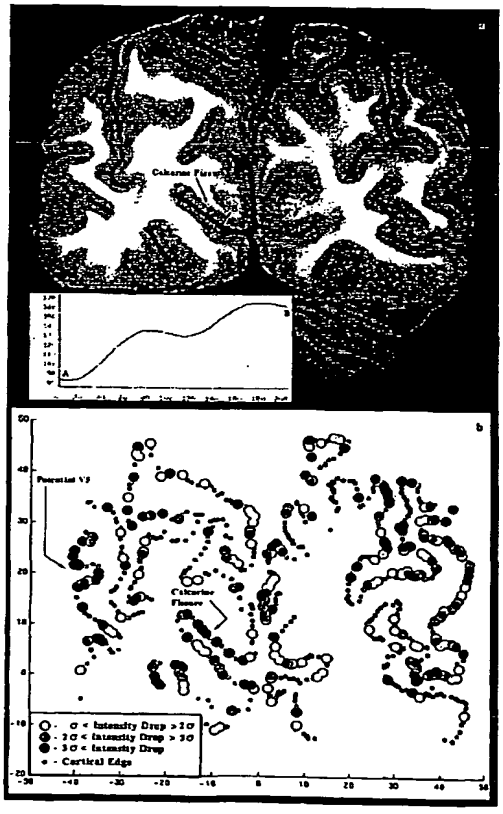
(75) Inventors/Applicants *(for US only)*: **WATSON, John, Douglas, Glenton** [AU/AU]; 142 Edinburgh Road, Castlecrag, New South Wales 2068 (AU). **EGAN, Gary, Francis** [AU/AU]; 120 Highland Road, North Keilor, Victoria 3036 (AU). **WALTERS, Nathan, Brett** [AU/AU]; 103/26 Kirketon Road, Darlinghurst, New South Wales 2010 (AU). **JENKINSON, Mark** [AU/AU]; 19 Davison Street, Brunswick, Victoria 3056 (AU). **BRAMLEY-MOORE, Mostyn** [AU/AU]; c/o Howard Florey Institute of Experimental Physiology and Medicine, University of Melbourne, Parkville, Victoria 3052 (AU).

(74) Agent: **GRIFFITH HACK**; GPO Box 4164, SYDNEY, New South Wales 2001 (AU).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH,

[Continued on next page]

(54) Title: A MAP OF A PROPERTY



(57) Abstract: The present invention provides a method of mapping a property of a three dimensional object. The method comprises the steps of mapping the property for at least a portion of the object, providing a line or region which defines intersections with part of the mapped portion, and displaying the property for the intersections. The present invention also provides another method of mapping a property of an object. The method comprises the steps of mapping the property for a slice within the object, providing a line within the slice, and displaying the property for the line.

WO 02/098292 A1

GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A MAP OF A PROPERTY

FIELD OF THE INVENTION

The present invention relates to a method of creating a map of a property of an object and to a map when created in accordance with the method. The invention relates particularly, though not exclusively, to a method of creating a map of a property derived from nuclear magnetic resonances related to a line or region within an object.

BACKGROUND OF THE INVENTION

It is currently of significant interest in different areas of research and analysis, in particular in medicine, to obtain non-destructively images or line-scans related to regions within objects. Typical objects may include biological specimen such as live human brains. The analysis of such images or line-scans may reveal information that is essential for understanding the functional organisation of the human brain.

In order to gain a more detailed understanding of the activation of the cerebral cortex of the brain, the anatomical organisation of the brain has to be known. With regard to the cerebral cortex, which is of particular interest in studies of cognition, a key question is how one relates the results of a functional neuro-imaging study with particular cortical areas in which changes in neural activity occur.

An enormous and costly effort is expended in carrying out functional neuro-imaging of ever increasing resolution, but in the end the researcher is left with only a crude

anatomical estimate of the activated cortical areas. If this situation is to be resolved, better human brain atlases are required. However, these suffer from one major problem; namely inter-subject variations in brain micro- and macrostructure.

Histological examination of the cyto-architecture, myelo-architecture and chemo-architecture of the brain remains the best method of accurately defining anatomically distinct regions. Ideally, if this can be performed in every subject used in functional neuroimaging studies, each individual's functional results can be precisely and accurately related with their own anatomical brain map. Furthermore, if a functionally active cortical focus were consistently found to be correlated with a well defined anatomical region across a population of subjects, this would significantly advance understanding of human brain organisation. Obviously, this is not practical except in a few rare cases. Instead, the results from postmortem brains of "non-subjects" in functional studies are used which again results in problems of inter-subject variability.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided a method of mapping a property of a three dimensional object, said method comprising the steps of

- mapping the property for at least a portion of the object,
- providing a line or region which defines intersections with part of said mapped portion, and
- displaying the property for the intersections.

According to a second aspect of the invention there is provided a method of mapping a property of a three dimensional object, said method comprising the steps of:

- mapping the property for a plurality of slices within the object,
- providing a line or region which defines intersections with part of at least some of the plurality of slices, and
- displaying the property for the intersections.

Preferably the property relates to nuclear magnetic resonances. More preferably, mapping of the property for the slices results in a plurality of nuclear magnetic resonance images.

It should be understood that nuclear magnetic resonances and nuclear magnetic resonance images are terms usually used by physicists and chemists. It is common practice to abbreviate these terms in a clinical environment to magnetic resonances and magnetic resonance images, respectively.

Preferably the line or region is a line. More preferably, the line is a plurality of lines.

Preferably, each of the lines is substantially perpendicular to the surface of the three-dimensional object.

In one embodiment the object is a biological object such as the brain of a subject (or another organ of a human or animal, or a semi-biological or inanimate object) and each of the lines is a line through the cortex of the brain. The cortex of the human brain is convoluted and

contains a plurality of grooves. Therefore, a line substantially perpendicular to the surface, for example within a groove, would typically intersect not only one slice, but can intersect a plurality of slices which are related to nuclear magnetic resonance images taken from adjacent areas of the brain.

Preferably, the method further comprises the steps of approximating the shape of the object by a three dimensional lattice of two-dimensional forms created by a first computer routine. More preferably, the lattice is a mesh.

Preferably, the lines are each substantially perpendicular to a respective one of the two-dimensional forms.

Preferably the intersections of each of the lines with each of the slices are selected using a second computer routine. More preferably a third computer routine maps nuclear magnetic resonance values for the intersections by displaying an array of values.

Preferably the method further comprises the step of analysing the array of values using a fourth computer routine. More preferably the fourth computer routine analyses quantitative properties.

Preferably analysing the array of values using the fourth computer routine comprises differentiation. More preferably analysing the array of values comprises determining the positions of the maxima, minima and/or zero-points of the array of values and the derivatives of the array of values.

In one embodiment, the positions of the inner and outer boundaries of the grey matter of the brain may be approximated by determining the positions of the maxima of the first derivative of the array of values. Two closely followed zero-points of the first derivative may indicate the presence of lamination. By measuring the relative positions of the zero points the thickness of the lamination can be determined.

According to a third aspect of the invention there is provided a map produced by a method of mapping a property of a three dimensional object, said method comprising the steps of:

- mapping the property for at least a portion of the object,
- providing a line or region which defines intersections with part of said mapped portion, and
- displaying the property for the intersections.

According to a fourth aspect of the invention there is provided a map produced by a method of mapping a property of a three dimensional object, said method comprising the steps of:

- mapping the property for a plurality of slices within the object,
- providing a line or region which defines intersections with part of at least some of the plurality of slices, and
- displaying the property for the intersections.

According to a fifth aspect of the invention there is provided a method of mapping a property of an object, said method comprising the steps of:

- mapping the property for a slice within the object,
- providing a line within the slice, and
- displaying the property for the line.

Preferably the property relates to nuclear magnetic resonances. More preferably, mapping of the property for the slice results in a nuclear magnetic resonance image.

Preferably, the line is a plurality of lines. More preferably in the fourth aspect a computer routine maps nuclear magnetic resonance values for each of the lines by displaying an array of values.

In one embodiment the object is a biological object such as a brain and each of the lines is a line through the cortex of the brain.

Preferably the method further comprises the step of analysing the array of values using a fifth computer routine. More preferably the fifth computer routine analyses numerical properties.

Preferably analysing the array of values using the fifth computer routine comprises differentiation. More preferably analysing the array of values comprises determining the positions of the maxima, minima and/or zero-points of the array of values and the derivatives of the array of values.

In one embodiment, the positions of the inner and outer boundaries of the grey matter of the brain may be approximated by determining the positions of the maxima of the first derivative of the array of values. Two closely followed zero-points of the first derivative may

indicate the presence of lamination. By measuring the relative positions of the zero points the thickness of the lamination can be determined.

According to a sixth aspect of the invention there is provided a map produced by a method of mapping a property of an object, said method comprising the steps of:

- mapping the property for a slice within the object,
- providing a line within the slice, and
- displaying the property for the line.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 (a) shows a nuclear magnetic resonance image of a living human brain. The insert shows a typical line scan. Figure 1 (b) shows lamination within the image shown in Figure 1 determined by a computer routine.

Figure 2 (i) shows a histological image of a section of a human brain (intensities changed), and the insert shows the original image. Figure 2 (ii) shows a nuclear magnetic resonance image corresponding to the histological image of Figure 2 (i). Figures 2 (iiia) and 2(iiib) show histological profiles as indicated in Figure 2 (i). Figure 2 (iv) shows intensity line scans corresponding to regions indicated in Figure 2(i) and Figure 2 (ii).

Figure 3 shows a mesh of two-dimensional forms that approximates the shape of a human brain.

Figure 4 shows the analysed nuclear magnetic resonance values in a three-dimensional visualisation. Darker regions indicate lamination.

Figure 5 shows a MRI slice of cartilage from a human knee and the insert shows line scans along which analyses were conducted.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Example 1: Two-dimensional Semi-automated Cortical Lamination Analysis

(Computer codes referred to throughout this example are listed in the Appendix.)

High-resolution T1 nuclear magnetic resonance images (NMRI) of the brain with high signal, low signal-to-noise, and high grey/white and grey/grey matter contrast suitable for lamination analysis were obtained. *In vivo* studies were performed using a magnetic field of 1.5 T and post-mortem studies were performed at 4.7 T.

Figure 1 (a) shows a nuclear magnetic resonance image of living human brain. Eight T1 weighted spin echo images were acquired on a GE Signa NMRI scanner in a single session using a 7.5" surface coil positioned at the occipital poles. These were aligned, re-sampled and averaged. This image shows evidence of the striate cortex within the calcarine fissure (arrow). [The parameters were: number of slices = 148; slice thickness = 0.700mm; field of view (FOV) = 16 cm^2; matrix size = 256 x 256; In-plane resolution = 0.625mm x 0.625mm; Echo Time (TE) = 2.7s; Repetition Time (TR) = 12.4s; Number of Excitations (NEX) = 1; Flip Angle = 25°; Inversion Time (TI) = 350ms].

The insert of Figure 2 (i) shows a histological image of a section of a human brain and the expanded view shows the same image with changed intensities. Figure 2 (ii) shows a nuclear magnetic resonance image corresponding to the histological image of Figure 2 (i). [The parameters were: number of slices: 30; slice thickness: 1mm (no gap); FOV: 8 x 4 cm2; matrix size: 512 x 256; in-plane resolution: 156µm x 156µm; TE_{eff} = 82.4 ms; TR = 6 s; NEX = 4; Acquisition Time (TA) = 1 hr 42 min 24 s]. Figures 2 (iiia) and 2(iiib) show histological profiles related to areas indicated in Figure 2 (i).

Once the images were acquired, they were analysed to determine cortical areas of lamination. This involves examining the intensity changes across the cortical grey matter in the images. It is known that the relative concentrations of myelin within cortical laminae can be directly measured in the striate cortex using high-resolution NMRI. Depending on the scanning protocol used, areas of myelination display either a dip or a peak in the intensity profile across that area of cortex.

To the analysis a series of intensity line profiles around the entire cortex are taken perpendicular to the cortical surface and through the grey matter. Figure 2 (iv) shows intensity line scans from corresponding regions of striate cortex in the histological slide 2(i) and the NMRI slice 2(ii) and the Insert of Figure 1(a) shows a typical line profile showing a characteristic intensity drop. ("A" corresponds to a point just outside the cortical surface whilst "B" corresponds to a point just deep of the grey/white matter junction).
Figure 1 (b) shows lamination determined using the computer routine "corla". This plot shows regions in

which there is an intensity dip found within the line profile indicating a particular pattern of cortical lamination. Results were thresholded and grouped based on a measure of the noise from the average image found by calculating the standard deviation of the signal intensity from an apparently uniform area (the background). Large pale grey circles are those in which the intensity drop is between 1 and 2 standard deviations, large dark grey circles between 2 and 3, and black circles greater than 3.

The program "Corla" implements a two-dimensional, slice-by-slice method and requires as input a series of contiguous evenly spaced line profiles taken through the cortical grey matter, perpendicular to its surface. The main steps taken in the analysis consist of determining:

- cortical boundaries: the air/cortical boundary and the grey/white matter boundary,
- the presence or absence of cortical lamination evidenced by an internal intensity drop (ie a local intensity peak followed by a trough) - if present, the number, position, thickness and intensity drop for the lamina are calculated.

The results are represented as an array of coordinates along the cortical boundary each of which has assigned to it a series of values which characterise the line profile at that point through the grey matter normal to the surface. These results are then graphically represented for ease of interpretation.

1. Line Profiles

The lines were selected through the cortex and spaced as evenly as possibly approximately 3-5 mm apart. Each line

was extended passed both cortical boundaries to ensure the entire grey matter was sampled. Using the line profile function in MEDx3.2, the line-profiles were recorded to file. The program to analyse these lineprofiles was written in MATLAB 6.0. The line profiles were read in as an array consisting of information about each line profile, namely the starting and finishing voxels, followed by a series of intensity readings.

2. Boundary Determination

This array is fed into a sub-program, "inflex", which determines the inner and outer grey matter boundaries. The boundaries are found by assuming they occur where there is a maximum change in intensity values between two adjacent points ie inflection points in the line profile or peaks in the first differential of this plot. There will an inflection point at the start and end of each line profile marking each boundary. An approximate first differential is taken of the line profile by calculating differences in intensity values between immediately adjacent points. The boundaries are related to the maximum peaks in this differential at the start and end of the plot. This function finds local maxima by using a moving window to smooth smaller fluctuations finds the midpoint of any plateaus. The size of the window determines the number of distinct local maxima it finds. The maximum peak in the first half of the plot is then set as one boundary and the maximum peak in the second half of the plot as the other. These positions, and their corresponding intensity values for all the line profiles are saved as an array and are the output of the sub-program "inflex".

3. Lamina Determination

The original array is also fed into a sub-program, "lamfinder", which determines if any lamination is present, the number of lamina, and various characteristics of the lamination. Lamination is detected by a dip in the intensity profile ie a local peak followed by trough. "Lamfinder" finds the peaks and troughs by finding where the approximated 1st differential crosses zero and where there is a change in sign on either side of this point in the first differential. Again, some smoothing is done to avoid minor non-significant stationary points and to find midpoints of plateaus. Once it has determined the position of the peaks and troughs and their corresponding intensity values, it then calculates:

- position of the peak and trough relative to the cortical boundaries as determined from "inflex".
- A measure of the thickness of the lamination (ie difference in position between peak and trough) relative to the cortical thickness.
- The intensity drop across the lamination (intensity difference between peak and trough) relative to the intensity drop across the entire grey matter (alternatively, relative to the intensity value at the outer cortical boundary)
- Using the output from "inflex", the sub-program "coord" calculates the actual (x,y) coordinates of the cortical boundaries. These are used for plotting the cortical edge to give graphical representation of the results.

4. Final Interpretation and Representation

Because even minor dips in intensity are detected by this program, it is necessary to threshold the results before plotting them. The intensity drops across the lamination have been used for this purpose. Discarding any "laminations" where the intensity drop is less than 5% of the total grey matter intensity drop clears a lot of the "noise" out of the results, and specific regions of lamination begin to appear. Higher thresholds yield more localised regions. A number of different aspects of the lamination can be plotted. Over a simple 2D plot of each slice showing one of the cortical edges the position of points of lamination can be plotted. Each point is colour coded to represent the degree of intensity drop found for that lineprofile. Multiple single slice 2D plots can then be stacked to give a 3D plot showing regions of lamination on the cortex. Regions with similar intensity drops can then be identified, as can a gradual change in the intensity drop which may mark a region with poorly defined borders.

Alternatively, using one of the cortical edges as a template, a 3D plot can be generated by projecting up from this the position of the start and finish of lamination found for that lineprofile at that point. This is useful in determining how the position and thickness of the lamination various around a single slice.

Automatic line profile generation and analysis

The following will describe an alternative procedure for the generation and analysis of line profiles. This process uses a software rountine which can broadly be divided into two parts: (A) Automatic line profile generation. The operation of the software is now outlined

in the context of an analysis of a brain. Line scans that were generated and along which analyses were conducted using this software routine are shown in Figure 5.

A. Automatic line profile generation (functions: Getbounds; skeleton; skelcontours; closestpt)

- calculate a skeletal representation of the grey matter of a brain (GM) from a segmented image (use distance transform method)
- the skeleton allows a set of non-intersecting profiles to be drawn across the GM
- these profiles are unevenly spaced and often have many share a single end point
- prune these profiles to keep only the shortest whenever they share an end point
- interpolate between the pruned profiles to get an even set of non-intersecting profiles

Function: GetBounds

- Start with the segmented image, with GM=1, WM=2, CSF/other=0
- calls on:

Function: skeleton

- Form a new image such that
  - (A) all GM points that border the with matter of a brain (WM) and let them=2
  - (B) also find all GM points that border CSF/other and let them=3
  - (C) let all other GM points=1
  - (D) and all remaining (non-GM) points=0

Function: skelcontours

- For all points that=1 in this new image find
  (using function: closestpt)
  (A) the minimum distance for a point of value 2
  (B) the minimum distance for a point of value 3
- Whenever the above two distances differ by a small amount (say sqrt(2) in 2D)then that point is marked as a skeleton point and the points of values 2 and 3 that are at the minimal distances are the end points of the profile.
- Prune this set by only keeping the shortest profile
  out of each set of profiles that share an end point.

B. Line profile analysis (Functions: TwoD; iprofile; IPA; coord; stationary; concavity; peaks)

Function: TwoD

- read in the coordinates of the line profiles generated by Getbounds
- user generated input read in
- calls on functions:

Function: iprofile

- generates interpolated line profiles between the specified coordinates

Function: IPA

- returns the coordinates of the stationary points of the line profile, widths,relative depth and intensity drops related to the stationary points (using function: stationary; peaks; coord)

- returns the coordinates of the concavity changes of the line profile (using function: concavity; peaks, coord)
- plots the line profiles if required

- Output is user specified results overlaid on original image using data returned: choices of results to view include position, depth, number, width and intensity drop of laminations.

Example 2: Three-dimensional Volumetric Automated Cortical Lamination Analysis

[Computer codes referred to throughout this example are listed in the Appendix. An off the shelf function "spm_hread" was used to read in the header information for the image (SPM99, freely available). All computer codes are in MATLAB 6.0].

The 3D analysis builds on the 2D analysis already described. In this example there are five steps in the 3D approach.

1. 3-D Representation of the Brain

   As line profiles need to be dropped perpendicularly through the cortex, the surface of the brain needs to be accurately characterised in order to get the correct topography.

   The surface was represented as a mesh, using the software package EMSE Suite (Source Signed Imaging, Inc.), although the analysis is not dependent on what program was used to generate the mesh, and can be

adapted to use others. EMSE approximates the surface as a mesh made up of triangles.

2. Information (meshes and images) are read in and normals to surfaces calculated

This module needs to read EMSE (or in the future other formats) meshes. It outputs lines along which the profiles need to be generated. Initially it has been agreed that the normals to the faces of the mesh would be calculated. This is due to coding and computational simplicity. The degree of sampling therefore depends on the size of the mesh. As the number of faces is increased, their size decreases and hence the density of the sampling will also increase. Additional normals at vertices, and further interpolation along the edges and/or faces can then be added incrementally so that ultimately, every point on the surface of the cortex can be sample. "Quickrun" is the initiating program, prompting the user to input information and choose analysis preferences. It calls the function "read_emse" which reads in the mesh. The functions "tricenter", "trinormal" and "normalize" then calculate the normals to each triangular face of the mesh through its centre. The function "snc" is then called which generates the line profiles and performs the analysis.

3. Use normals and original Analyze image to calculate intensity line profiles

This is the function of "snc". It reads in the image data using "anzload" and, using the normals calculated previously, generates the line profiles. The function "lpa" is then called which performs the actual line profile analysis.

4. Perform cortical lamination analysis on line profiles

"Lpa" determines changes in concavity stationary points and boundary points by calling various functions. The function "concavity" finds the changes in concavity along the line profile. These correspond to where the second differential crosses zero. Therefore, it passes the first differential of the line profile to the function "peaks" which then finds the differential of this (ie it uses the second differential in this case). The zero points are found by determining where there is a change in sign along this curve. The function ""boundary_pts" then uses the information from "concavity" to determine the cortical boundaries using a vector method which incorporates a number of competing criteria for assigning the cortical boundaries. The function "stationary" determines stationary points in the line profile. These correspond to where the first differential crosses zero. Therefore, "stationary" passes the line profile to "peaks" which then finds the zero points in the first differential in the same way it did for "concavity". Using all this information, "lpa" then determines a number of features of each line profile such as the number of laminations, their widths and the intensity drops corresponding to each.

All data is thresholded using an intensity change significance threshold and a resolution significance threshold. The former is determined from the signal-to-noise ratio of the original image whilst the latter is determined from the resolution of the original image.

5. Visualisation of results

The information from "lpa" is used by "snc" to colour the mesh using the function "colour_picker"

according to what results the user wishes to visualize. A colour coded wireframe in OOGL (freeware: suitable for geomview) is generated. OOGL can be easily converted to VRML using available tools. This produces a 3D model which can be rotated and rendered in real time using geomview (freeware). The colour represents a particular analysis result and the intensity of that colour quantitatively reflects the magnitude of that result. Fanother option may involve overlaying the MR and the wireframe and also combining multiple coloured meshes representing various combinations of results. Similarly coloured regions would represent cortical regions sharing a common lamination feature eg, intensity drops, positions of laminae, thicknesses, number of laminae etc. Figure 3 shows a mesh of two-dimensional forms that approximates the human brain. Darker regions correspond to darker shades of colour and indicate, in this example, lamination. Figure 4 shows the corresponding three-dimensional visualisation without the mesh.

It will be appreciated that each line profile contains a large amount of information about the cortex through which it passes in the MRI image. In this example it has been chosen to examine one aspect of that by using first and second differentials to determine stationary points and changes in concavity. As the quality and or type of images improves, or even regardless of this, there are many other ways of examining and analysing the information that can be incorporated into this analysis which can aid in defining distinct regions.

It will also be appreciated that this method is not limited to biological objects. The method of mapping the

property for the object is not limited to the detection of a mapping the property for a plurality of slices. Alternative methods can be used which may not comprise the detection of slices. It will also be appreciated that the method is not limited to nuclear magnetic resonance imaging. Each of the lines may also be a region. Alternative computer routines may be used to analyze the array of values which may or may not comprise numerical procedures such as differentiation.

It will also be appreciated that a procedure similar to the automatic line profile generation and analysis described in the context of 2-D analysis can be extended to be applicable 3-D analysis.

Although the invention has been described with reference to particular examples, it will be appreciated by those skilled in the art that the invention may be embodied in many other forms.

It is to be understood that, if any prior art publication is referred to herein, such reference does not constitute an admission that the publication forms a part of the common general knowledge in the art, in Australia or any other country.

Appendix

Codes of computer routines referred to throughout the examples

Computer code used for 2-D analysis

```
function [Lam,E,LamData,Idrop]=Corla(t1,t2,fig,Z)
[lpStart,lpFinish,x]=reader3;
[r,n]=size(x);
y=([1:r]');
LamData=[];
for i=1:n
  z=[y,x(:,i)];
  t=diff(z(:,2));
  [E(i).ce,E(i).gwb]=inflex(z);
  [Lam(i).peak,Lam(i).trough,LamD,Id]=lamfinder(z(:,2),E(i).ce(1),E(i).gwb(1));
  if isempty(LamD)
    LamData(:,:,i)=zeros(30,3);
    Idrop(:,:,i)=zeros(30,3);
  else
    if length(LamD(:,1))~=30
      Id(30,3)=0;
      LamD(30,3)=0;
    end
    LamData(:,:,i)=LamD;
    Idrop(:,:,i)=Id;
  end
  Profileplotter(z,E(i),Lam(i),i);
end
[Start,Finish]=coord(LamData,lpStart,lpFinish,E,t1,t2,fig,Idrop,Z);
```

```
function [Start,Finish]=coord(LamData,lpStart,lpFinish,E,t1,t2,fig,Idrop,Z)
Start=[];
Finish=[];
figure(fig);
hold on
for i=1:length(lpStart)
  [S,F]=pointplot(lpStart(i,:),lpFinish(i,:),E(i));
  Start=[Start;S];
  Finish=[Finish;F];
  plot(Start(i,1),Start(i,2),'bo');
  hold on
end
h1=findobj(gcf,'Type','line');
```

```
set(h1(:),'MarkerSize',3.5);
set(h1(:),'LineWidth',3.5);
for i=1:length(lpStart)
  if (~isempty(find(Idrop(:,3,i)>=t1)))
    plot(Start(i,1),Start(i,2),'ro');
    hold on
    h1=findobj(gcf,'Type','line');
    set(h1(1),'LineWidth',7.5);
  end
end
for i=1:length(lpStart)
  if (~isempty(find(Idrop(:,3,i)>=t1)))
    plot(Start(i,1),Start(i,2),'go');
    hold on
    h1=findobj(gcf,'Type','line');
       set(h1(1),'MarkerSize',5.5);
    set(h1(1),'LineWidth',5.5);
  end
end
for i=1:length(lpStart)
  if (~isempty(find(Idrop(:,3,i)>=(2*t1))))
    plot(Start(i,1),Start(i,2),'yo');
    hold on
    h1=findobj(gcf,'Type','line');
       set(h1(1),'MarkerSize',5.5);
    set(h1(1),'LineWidth',5.5);
  end
end
for i=1:length(lpStart)
  if (~isempty(find(Idrop(:,3,i)>=(3*t1))))
    plot(Start(i,1),Start(i,2),'ro');
    hold on
    h1=findobj(gcf,'Type','line');
       set(h1(1),'MarkerSize',5.5);
    set(h1(1),'LineWidth',5.5);
  end
end

figure(fig+1);
hold on
for i=1:length(lpStart)
  t=max((Idrop(:,3,i)));
  v=find(Idrop(:,3,i)==t);
  if t>3
          plot3(Start(i,1),Start(i,2),LamData(v(1),2,i),'b*');
    hold on
    plot3(Start(i,1),Start(i,2),LamData(v(1),1,i),'r*');
    hold on
  end
end
```

```
%g=1;
   %     while (g<=length(LamData(:,1,i)))&(LamData(g,1,i)>0)
  %      if (LamData(g,2,i)-LamData(g,1,i))>5
   %           plot3(Start(i,1),Start(i,2),LamData(g,2,i),'b*');
   %           hold on
  %plot3(Start(i,1),Start(i,2),LamData(g,1,i),'r*');
   %     hold on
   %     end
   %     g=g+1;
    %    end
%else
   %     plot3(Start(i,1),Start(i,2),Z,'bo');
 %     hold on
```

```
function [S,F]=pointplot(lpStart,lpFinish,E)

S(1,1)=lpStart(1,1)+(E.ce(1)./200)*(lpFinish(1,1)-lpStart(1,1));
S(1,2)=lpStart(1,2)+(E.ce(1)/200)*(lpFinish(1,2)-lpStart(1,2));
F(1,1)=lpStart(1,1)+(E.gwb(1)/200)*(lpFinish(1,1)-lpStart(1,1));
F(1,2)=lpStart(1,2)+(E.gwb(1)/200)*(lpFinish(1,2)-lpStart(1,2));
```

```
function [PPos,Pheight]=avg(lp,x,PPos)
tsum=0;
for i=1:length(x)
  if PPos<8
    tsum=tsum+x(i);
  else
    tsum=tsum+(x(i)-9);
  end
end
PPos=round(((PPos*length(x))+tsum)/length(x));
Pheight=lp(PPos);
```

```
function [ce,gwb]=inflex(lp)
x=diff(lp(:,2));
n=length(x);
P=[1,x(1)];
count=0;
i=1;
d=1;
inf=[];
```

```
I=[];
while i<n-10
  bit=x(i:i+9);
  localmax=max(bit);
     y=find(bit>(localmax-1));
     temp=size(y);
     if count<=1
          d=y(temp(1,2));
     else
          d=y(1);
     end
  if x(d+i)>P(1,2)
          P=[d+i,x(d+i)];
     else
          if count==0
               P=refiner(x,P,n);
                         inf=[inf;P];
               count=count+1;
        elseif (d+i)-inf(count,1)>10
          P=refiner(x,P,n);
          inf=[inf;P];
          count=count+1;
        end
          P=[d+i,x(d+i)];
     end
     i=i+1;
end
a=find(inf(:,1)<100);
[ceI,ceP]=max(inf(1:length(a),2));
ce=[inf(ceP,1),ceI];
[gwbI,gwbP]=max(inf(length(a)+1:count,2));
gwb=[inf((gwbP+length(a)),1),gwbI];
I=[lp(inf(:,1)),inf(:,2)];
```

```
function [Peak,Trough,LamData,Idrop]=lamfinder(lp,start,finish)
if finish<length(lp);
  n=finish;
else
  n=length(lp);
end
count=0;
temp=0;
i=20;

f=finish;
x=diff(lp);
peakp=[];
```

```
peakI=[];
troughp=[];
troughI=[];
while i<(n(1)-31)
  if (x(i))<=0
    temp=i;
    while (x(temp)==0)&(temp<(n(1)-31))
      temp=temp+1;
    end
    if x(temp)<0
          count=count+1;
          peakp=[peakp;(temp+i)/2];
               peakI=[peakI;lp(temp)];
          while (x(temp)<0)&(temp<(n(1)-31))
               temp=temp+1;
          end
          i=temp;
          if x(temp)==0
               while (x(temp)==0)&(temp<(n(1)-31))
               temp=temp+1;
               end
          end
     troughp=[troughp;(i+temp)/2];
     troughI=[troughI;lp(i)];
     else
      i=temp+1;
    end
  else
    i=i+1;
  end
end
Peak=[peakp,peakI];
Trough=[troughp,troughI];
Pdata=[];
Tdata=[];
Idrop1=[];
Idrop2=[];
if ~isempty(Peak)
  Pdata=((Peak(:,1)-start)*100)/(finish-start);
  Tdata=((Trough(:,1)-start)*100)/(finish-start);
  LamData=[Pdata,Tdata];
  Idrop3=(Peak(:,2)-Trough(:,2));
  Idrop1=((Peak(:,2)-Trough(:,2))*100)/(lp(finish)-lp(start));
  Idrop2=((Peak(:,2)-Trough(:,2))*100)/lp(start);
  Idrop=[Idrop1,Idrop2,Idrop3];
else
  LamData=[];
  Idrop=[];
end
```

```
function Profileplotter(z,E,Lam,i);
figure(i);
plot(z(:,1),z(:,2),'b-',1:199,50*(diff(z(:,2))),'g-');
grid on
hold on
plot((E.ce(1)),(E.ce(2)),'ro',(E.gwb(1)),(E.gwb(2)),'ro')
hold on
if ~isempty(Lam.peak)
  plot((Lam.peak(1,1)),(Lam.peak(1,2)),'ro');
  plot((Lam.trough(1,1)),(Lam.trough(1,2)),'ro');
end
```

```
function [Start,Finish,mat]=reader
Start=[];
Finish=[];
mat=[];
h=dir('106-7*');
n=length(h);
for i=1:n
  %s=h(i).name
  fid=fopen(h(i).name,'rt');
  for j=1:2
    fgetl(fid);
  end
  crap=fscanf(fid,'%s\t%s\t%s\t,');
  S=(fscanf(fid,'%f,'))';
  fgetl(fid);
  crap=fscanf(fid,'%s\t%s\t%s\t,');
  F=(fscanf(fid,'%f,'))';
  Start=[Start;S];
  Finish=[Finish;F];
  for j=1:6;
    fgetl(fid);
  end
  temp=[];
  for j=1:200
    data=fscanf(fid,'%i\t%i,');
    fgetl(fid);
    temp=[temp;data(2,1)];
  end
  %temp2=[];
  %for j=1:200
   %  temp2=[temp2;temp(201-j)];
  %end
```

```
  mat=[mat,temp];
  fclose(fid);
end
```

```
function Peak=refiner(lp,P,n)
PPos=P(1);
Pheight=lp(P(1));
if (PPos>8)&(PPos+8<n)
      [Pheight,temp]=max(lp(PPos-8:PPos+8));
  PPos=temp-9+PPos;
elseif PPos<=8
  [Pheight,PPos]=max(lp(1:PPos+8));
else
  [Pheight,temp]=max(lp(PPos-8:n));
  PPos=temp-9+PPos;
end
if (PPos>8)&(PPos+8<n)
  x=find(lp((PPos-8:PPos+8))>Pheight-3);
  [PPos,Pheight]=avg(lp,x,PPos);
elseif PPos<=8
  x=find(lp((1:PPos+8))>Pheight-3);
  [PPos,Pheight]=avg(lp,x,PPos);
else
  x=find(lp((PPos-8:n))>Pheight-3);
  [PPos,Pheight]=avg(lp,x,PPos);
end;

Peak=[PPos,Pheight];
```

Alternative computercode used for 2-D analysis

```
function [im0, coords] = Getbounds

nl = sprintf('\n');
disp('Please enter the name of the file to be analysed');
Image= deblank(input('Image file: ','s'));
disp(nl);
disp('Please enter the name of the segmented grey matter file');
Grey=deblank(input('Grey matter image file: ','s'));
disp(nl);
disp('Enter 1 if the file is avw, and 0 if it is ppm');
ftype = input('file type: ');
disp(nl);
disp('Do you want to subsample the image?');
disp('(type 1 if you do not, otherwise the factor to downsample by
(integer greater than 1))');
sf = input('subsample factor: ');
disp(nl);
```

```
disp('Please enter the name of the output file for the
coordinates?');
base = deblank(input('file name: ','s'));
disp(nl);

if ftype==0,
      im0 = readppm(Image);
      gm = readppm(Grey);
elseif ftype==1,
      im0=read_avw(Image);      % read original image
      gm=read_avw(Grey);        %read grey matter image
end

im0 = im0(:,:,1);
gm = gm(:,:,1);
figure(1);
imagesc(im0);
colormap(gray);
hold on;

% now make the segmented image (WM=2, GM=1, other=0)
segim = (im0>1)*2 - (gm>1);

% subsample the image (factor of "sf")
if sf > 1
      [N,M]=size(segim);
      segim = segim(1:sf:N,1:sf:M);
end

figure(2)
imagesc(gm);
colormap(gray);
pause

figure(3)
imagesc(segim)
colormap(gray);
pause
% fix up any problems (I had to fill a couple of holes in the GM)
% now generate the coordinates of the unique profiles
coords = getcontours(segim);

% convert coordinates back to original if necessary (upsample)
if sf > 1
      coords = (coords-1)*sf + 1;
end

% write the coordinates to file
output_file = strcat(base, '.dat');
fout = fopen(output_file,'w');
for i=1:length(coords)
      fprintf(fout,'%f %f %f %f\n', coords(i,1), coords(i,2),
coords(i,3),      coords(i,4));
end
fclose(fout);
```

```
function [dims,scales,bpp,endian,datatype] = read_avw_hdr(fname)
% [dims,scales,bpp,endian] = READ_AVW_HDR(fname)
%
```

```
%  Extracts the 4 dimensions (dims),
%   4 scales (scales) and bytes per pixel (bpp) for voxels
%   contained in the Analyse header file (fname)
%   Also returns endian = 'l' for little-endian or 'b' for big-endian
%
%  See also: READ_AVW, READ_AVW_IMG, SAVE_AVW, SAVE_AVW_HDR,
SAVE_AVW_IMG

% remove extension if it exists
if ( (length(findstr(fname,'.hdr'))>0) | ...
      (length(findstr(fname,'.img')>0)) ),
  fname=fname(1:(length(fname)-4));
end
fnhdr=strcat(fname,'.hdr');

% open file in big-endian
endian='b';
fid=fopen(fnhdr,'r','b');
testval = fread(fid,1,'int32');
% check if this gives the correct header size - if not use little-
endian
if (testval~=348),
  fclose(fid);
  fid=fopen(fnhdr,'r','l');
  endian='l';
  testval = fread(fid,1,'int32');
  if (testval~=348),
    disp('Can not read this file format');
    return;
  end
end
      % ditch the remaining initial header stuff
  dummy=fread(fid,36,'char');
      % ditch dim[0] = No. dimensions
  dummy=fread(fid,1,'int16');
  dims=fread(fid,4,'int16');
  dummy=fread(fid,3,'int16');
  dummy=fread(fid,14,'char');
  datatype=fread(fid,1,'int16');
  bpp=fread(fid,1,'int16');
  dummy=fread(fid,2,'char');
  dummy=fread(fid,1,'float');
  scales=fread(fid,4,'float');
fclose(fid);
return;
```

```
function img = read_avw_img(filename);
%  [img] = READ_AVW_IMG(filename)
%
%  Read in an analyse file into either a 3D or 4D
%   array (depending on the header information)
%  Ouput coordinates are in MEDx convention
%   except that all dimensions start at 1 rather than 0
%  Note: automatically detects char, short, long or double formats
%
%  See also: READ_AVW, READ_AVW_HDR, SAVE_AVW, SAVE_AVW_HDR,
SAVE_AVW_IMG

% remove extension if it exists
if ( (length(findstr(filename,'.hdr'))>0) | ...
```

```
      (length(findstr(filename,'.img')>0)) ),
  filename=filename(1:(length(filename)-4));
end
fnimg=strcat(filename,'.img');
fnhdr=strcat(filename,'.hdr');

[dims,scales,bpp,endian,datatype] = read_avw_hdr(fnhdr);
fp=fopen(fnimg,'r',endian);
if (datatype==4),
  dat=fread(fp,'short');
elseif (datatype==2),
  dat=fread(fp,'char');
elseif (datatype==8),
  dat=fread(fp,'int');
elseif (datatype==64),
  dat=fread(fp,'double');
elseif (datatype==16),
   dat=fread(fp,'float32');
end
fclose(fp);
nvox = prod(dims);
if (length(dat)<nvox),
  error('Cannot open image as .img file does not contain as many
voxels as the .hdr specifies');
elseif (length(dat)>nvox),
  disp('WARNING::truncating .img data as it contains more voxels than
specified in the .hdr');
  dat = dat(1:nvox);
end

if (dims(4)>1),
  img = reshape(dat,dims(1),dims(2),dims(3),dims(4));
else
  img = reshape(dat,dims(1),dims(2),dims(3));
end

clear dat;

%% DEFUNCT FLIPPING
%% flip y dimension to be consistent with MEDx
%img=flipdim(img,2);
```

```
function [coords]=getcontours(segim)
% coords = getcontours(segim)
%
% Input (segim) is a segmented image (2D) which has
%  values of 1 for GM, 2 for WM and 0 otherwise
% Output (coords) is a matrix where each row contains the
%  end coordinates of a cortical profile (4 values)
% The intensity profile can be extracted using
%  the iprofile() function with a given row from coords
%
% See also IPROFILE, SKELETON, SKELCONTOURS

im=conv2(segim,ones(3,3)/9,'same');
im2=(im<0.9).*(segim==1);
im3=(im>1.1).*(segim==1);
im=(segim==1)+2*im2+im3;
sk=skeleton(im);
coords=skelcontours(im,sk);
```

```
function [sk]=skeleton(im)
% sk=skeleton(im)
%
% Calculates the skeleton of an image im (medial axis transform)
% The image im is a binary image of the area with voxels at
%  one boundary set to 2 and at the opposite boundary set to 3
%  e.g. GM/WM interface = 2, GM/CSF interface = 3, GM = 1

[N M]=size(im);
skd=zeros(N,M,2);
skd(:,:,1) = im*0 + max(M^2,N^2)+1;
skd(:,:,2) = im*0 + max(M^2,N^2)+1;

% Calculate the minimum distance transform
xim=kron((1:N)',ones(1,M));
yim=kron(1:M,ones(N,1));
val=[2 3];
for r=1:2,
  for x=1:N,
    for y=1:M,
      if (im(x,y)==val(r)),
      dist = (xim-xim(x,y)).^2 + (yim-yim(x,y)).^2;
      skd(:,:,r) = min(skd(:,:,r),dist);
      end
    end
    disp(x)
  end
end
sk = (abs(sqrt(skd(:,:,1))-sqrt(skd(:,:,2)))<sqrt(2)).*(im==1);
```

```
function [coords]=skelcontours(im,sk,cand)
% [coords]=skelcontours(im,sk)
%
% Calculates the *unique* line profile contours from a skeleton (sk)
%  of an image im (medial axis transform) - as obtained from
%  skeleton()
% Returns the coordinates as an N by 4 matrix - each row having
%  the form [x1 y1 x2 y2]
% The image im is a binary image of the area with voxels at
%  one boundary set to 2 and at the opposite boundary set to 3
%  e.g. GM/WM interface = 2, GM/CSF interface = 3, GM = 1
%
% See also SKELETON

[sx,sy] = find(sk==1);
N = length(sx);

% get candidate coord pairs
if (nargin<3),
  cand=[];
  im2=(im==2);
  im3=(im==3);
  disp(N);
  for k=1:N,
    cand=[ cand ; closestpt(im2,[sx(k) sy(k)]) closestpt(im3,[sx(k)
...
                sy(k)]) ];
    disp(k);
```

```
  end
end

% eliminate multiple matches
for r=1:2,  % r=1 => boundary with im==2,   r=2 => bdy with im==3
coords=[];
for k=1:length(cand(:,1)),
  if (min(cand(k,:))>0),  % only do valid end points
    diff=abs(cand(:,(r*2-1))-cand(k,(r*2-1))) + abs(cand(:,r*2)-
cand(k,r*2));
    idx=find(diff==0);   % all rows with same end point
    mindiff = N*2;
    minidx = 0;
    mincand=[0 0 0 0];
    % find minimum contour with this end point
    for m=1:length(idx),
      d = norm([cand(idx(m),3)-cand(idx(m),1),cand(idx(m),4)-
cand(idx(m),2)]);
      if (mindiff>d),
      mindiff = d;
      minidx = idx(m);
      mincand = cand(minidx,:);
      end
      cand(idx(m),:)=[0 0 0 0];
    end
    if (minidx>0),
      if (min(mincand)>0),
      coords=[coords; mincand];
      end
    else
      error('FAILED ASSERTION: minidx>0');
    end
    disp(k);
  end
end
cand = coords;
end
```

```
function [xc]=closestpt(im,x)
% xc=closestpt(im,x)
%
% Calculates the closest voxel coordinate to x where the
%  binary image is 1.  (i.e. arg min_{xc} norm(x-xc) st im(xc)=1)

[N M]=size(im);
% Calculate the minimum distance transform
maxd=(M+N+1)^2;
xim=kron((1:N)',ones(1,M));
yim=kron(1:M,ones(N,1));
dist = im.*((xim-x(1)).^2 + (yim-x(2)).^2) + maxd*(1-im);
[cx,cy]=find(dist==min3(dist));
xc(1)=cx(1);
xc(2)=cy(1);
```

```
%%%%%%%%%%%%%%%%%
function TwoD(im0)
%%%%%%%%%%%%%%%%%
```

```
nl = sprintf('\n');
disp('Please enter the name of a coords file');
fin = deblank(input('coords file: ','s'))
disp(nl);
disp('Standard deviation of background noise?');
disp('(Enter the value 1, to ignore signifigance rel. to background
noise.)');
std_dev = input('std. deviation: ');
disp(nl);
disp('Enter value for number of points to take per voxel');
pspacing = input('spacing: ');
disp(nl);
disp('Do you want to display the line profiles?');
disp('(type 0 if you do not, any other number if you do.)');
pl = input('display profiles: ');
disp(nl);
disp('If you want to plot the number of laminae, enter 0');
disp('If you want to plot the position and intensity drop of the
myelinated laminae, enter 1');
disp('If you want to plot a combination of number of laminae with the
position, and intensity drops of the myelinated laminae, enter 2');
disp('If you want to plot the relative centre of each myelinated
laminae, enter 3');
result = input('option: ');
disp(nl)

fid = fopen(fin);
line = fgetl(fid);
C_num=0;
while line ~= -1,
      C = sscanf(line, '%f %f %f %f')';
      line = fgetl(fid);
      C_num=C_num+1;
      coords(C_num,:)=[C];
end
fclose(fid);
[t,u]=size(coords);
figure(10);
imagesc(im0);
colormap(gray);
hold on;

if result==0
for i= 1:t
      ip = iprofile(coords(i,:),im0,pspacing);
      [Ledges, Lcentres, widths, num_lam, Id,W,relW]=
IPA(coords(i,:),ip', std_dev, pl, pspacing,(i+1));
      figure(1)
      if num_lam <=1
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'r-o');
      elseif num_lam <=2
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'y-o');
      elseif num_lam <=4
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'g-o');
      elseif num_lam <=6
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'c-o');
      else
```

```
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'b-o');
      end
      hold on;
end

elseif result == 1
for i= 1:t
      ip = iprofile(coords(i,:),im0,pspacing);
      [Ledges, Lcentres, num_lam, Id,pcount,W,relW]=
IPA(coords(i,:),ip', std_dev, pl, pspacing,(i+3));
      figure(1)
      plot([coords(i,2) coords(i,4)],[coords(i,1) coords(i,3)],'r-');
      hold on;

      for j=1:pcount
            if Id(j,2) < 1
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'r+');
            elseif Id(j,2) < 2
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'y+');
            elseif Id(j,2) < 3
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'y*');
            elseif Id(j,2) < 4
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'g+');
            elseif Id(j,2) < 5
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'g*');
            elseif Id(j,2) < 6
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'c+');
            elseif Id(j,2) < 7
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'c*');
            elseif Id(j,2) < 8
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'b+');
            elseif Id(j,2) < 10
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'b*');
            else
                  plot(Lcentres(Id(j,1),2),Lcentres(Id(j,1),1),'m+');

            hold on;
            end
      end
end

elseif result==2
for i= 1:t
      ip = iprofile(coords(i,:),im0,pspacing);
      [Ledges, Lcentres, num_lam, Id,pcount,W,relW]=
IPA(coords(i,:),ip', std_dev, pl, pspacing,(i+3));
      figure(1)
      if num_lam <=1
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'b-');
      elseif num_lam <=2
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'g-');
      elseif num_lam <=4
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'c-');
      elseif num_lam > 4
            plot([coords(i,2) coords(i,4)],[coords(i,1)
coords(i,3)],'w-.');
      end
```

```
      hold on;
      if  num_lam <=4
      for j=1:pcount
            if Id(j,2) < 4 & Id(j,2) > 1
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'r*');
                  hold on;
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'ro');
            elseif Id(j,2) < 7
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'y*');
                  hold on;
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'yo');
            elseif Id(j,2) < 15
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'w*');
                  hold on;
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'wo');
            elseif Id(j,2) >=15
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'w*');
                  hold on;
                  plot(Lcentres(W(j,1),2),Lcentres(W(j,1),1),'ws');
            end
      hold on;
      end
      end
end

elseif result==3
for i= 1:t
      ip = iprofile(coords(i,:),im0,pspacing);
      [Ledges, Lcentres, num_lam, Id,pcount,W,relW]=
IPA(coords(i,:),ip', std_dev, pl, pspacing,(i+3));
      figure(10)
      plot([coords(i,2) coords(i,4)],[coords(i,1) coords(i,3)],'r-');
      hold on;
      if  num_lam <=4
      for j=1:pcount
            if relW(j,2) <= 0.15

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'m*');

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'mo');
            elseif relW(j,2) <= 0.325

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'b*');

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'bo');

            elseif relW(j,2) <= 0.5

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'c*');

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'co');
            elseif relW(j,2) <= 0.675

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'g*');

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'go');

            elseif relW(j,2) <= 0.85

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'y*');
```

```
      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'yo');

            else

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'w*');

      plot(Lcentres(relW(j,1),2),Lcentres(relW(j,1),1),'wo');

            end
            hold on;
      end
      end
end
end
```

```
function [ivals]=iprofile(coords,im,pspacing)
% [ivals] = iprofile(coords,im)
%
% Input (coords) is a matrix where each row contains a
%  2D coordinate pair (4 values) that specify the end
%  points of a cortical line profile
% Returns the intensity values at 1 voxel spacing (equiv)
%  If more than one profile is requested, the output rows
%  may be padded with zeros (to the right).
%  im is the image file and pspacing is the number of points read per
voxel
%
% See also GETCONTOURS, SKELETON, SKELCONTOURS

X=[];
maxlen=0;
for n=1:length(coords(:,1)),
  x0=coords(n,1:2);
  x1=coords(n,3:4);
  maxlen=max(maxlen,norm(x1-x0)*pspacing);
end

ivals = zeros([length(coords(:,1)),ceil(maxlen+1)]);

for n=1:length(coords(:,1)),
  x0=coords(n,1:2);
  x1=coords(n,3:4);
  len=norm(x1-x0)*pspacing;
  for r=0:len,
    x=x0+(x1-x0)*(r/len);
    ivals(n,r+1) = interp2(im,x(2),x(1),'linear');  %% swap x & y
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
function [Ledges, Lcentres, num_lam, Id, pcount,W,relW] =
IPA(coords,ip, std_dev, pl, pspacing,plotnumber);
% analyzes a line profile
%
% INPUT:
% ip is a 2xN matrix containing the interpolated line profile
```

```
% std_dev is the standard deviation of the background noise
% pl (optional) if pl > 0 then plot the line profile
%
% OUTPUT:
% Id is a list of the intensity drops of the lamina
% W and relW are a list of the widths of the lamina and the
coordinates for
% relative depth of the middle of the laminae
% num_lam and pcount are the number of lamina and number of peaks
respectively
% Ledges and Lcentres are lists of the coordinates for the concavity
changes and
% stationary points respectively
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

Lcentres=[];
Ledges=[];
c_count=0;
[x,y] = size(ip);

% find stationary points
[s_xs, s_ys, s_count, s_sign, Id, pcount,W,relW] = stationary(ip,
pspacing, std_dev);
if s_xs > 0
      Lcentres=coord(coords, s_xs, pspacing);
end

count lamina
num_lam = s_count;

% find changes in concavity
[c_xs, c_ys, c_count, c_sign] = concavity(ip, std_dev, s_xs, s_sign,
s_count, pspacing)
Ledges=coord(coords, c_xs, pspacing)

if pl <= 0, return; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
      % plot the line profile

scale1st_diff = 20; % magnitude of the gradient is small
scale2nd_diff = 30;
figure(plotnumber)
hold on;

dip = gradient(ip);
ddip = gradient(dip);
plot(1:x, dip.*scale1st_diff,'r');
hold on;
plot(1:x, ddip.*scale2nd_diff,'g')
hold on;
plot(1:x,ip(:,1),'b');
hold on;

t=1;
while t<= c_count
      % plot concavities
      if c_count > 0,
```

```
            plot(c_xs(t), ip(c_xs(t),1),'r+');
            hold on;
      end
      t=t+1;
end

t=1;
while t<= s_count
      % plot stationary points
      if s_count > 0,
            plot(s_xs(t), ip(s_xs(t),1),'g+');
      end
      t=t+1;
end

grid on;
set the scale for the line profile graphs
xmin = 0;
xmax = x;
ymin = -1000;
ymax = max(ip(:,1)) * 1.4;
axis([xmin, xmax, ymin, ymax]);
xlabel('distance (mm)');
ylabel('intensity');

hold off;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
function [xs, ys, count, sign, Id,pcount,W, relW] = stationary(ip,
pspacing, std_dev);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%

if nargin < 2 | nargin > 3
      error('need a vector containing the intensities along a line
profile & a search starting point (optional)');
end

W=[];
relW=[];
dip = gradient(ip);
[x,y] = size(ip);
count = 0;
xs(1) = -1;
ys(1) = -1;
rsign = 0;
sign = 0;
Id=[];
if y < 0, return; end

[rxs, rys, rcount, rsign]=peaks(ip, std_dev);
temp=1;
i=1;

while i<=rcount
      if rcount-i <=2 & rcount-i >= 0
            xs(count+1)=rxs(i);
            ys(count+1)=rys(i);
            sign(count+1)=rsign(i);
```

```
            count=count +1;
            i=i+1;
      else
            i=i+2;
            if rxs(i)-rxs(temp) > 0.5*pspacing
                  xs(count+1)=rxs(temp);
                  ys(count+1)=rys(temp);
                  sign(count+1)=rsign(temp);
                  xs(count+2)=rxs(i-1);
                  ys(count+2)=rys(i-1);
                  sign(count+2)=rsign(i-1);
                  count = count +2;
                  temp=i;
            else
                  if rsign(temp)*ip(rxs(temp))<ip(rxs(i))
                        temp=i;
                  end
            end
      end
end
pcount=0;
if xs <0 return; end

for j=1:length(xs)
      if sign(j) > 0
            %drop=ip(xs(j));
            if length(xs)==1 | j==1
                  drop= 100-100*ip(1)/ip(xs(j));
            elseif j<length(xs)
                  drop=100-100*abs((ip(xs(j-
1))+ip(xs(j+1)))/2)/ip(xs(j));
            else
                  drop=100-100*ip(xs(j-1))/ip(xs(j));
            end
            Id=[Id;j,drop];
            pcount=pcount+1;
      end
end

for k=1:length(xs)
      if sign(k) > 0
            if length(xs)==1
                  width=0.5*length(ip)/length(ip);
            elseif k==1
                  width=0.5*(xs(k+1)-1)/length(ip);
            elseif k<length(xs)
                  width=0.5*(xs(k+1)-xs(k-1))/length(ip);
            else
                  width=(0.5*(xs(k)-xs(k-1)) + length(ip)-
xs(k))/length(ip);
            end
            W=[W;k,width];
      end
end

for m=1:length(xs)
      if sign(m) > 0
            rw=xs(m)/length(ip);
            relW=[relW;m,rw];
      end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
function [xs, ys, count, sign] = concavity(ip, std_dev, pspacing);%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

if nargin < 3 | nargin > 4
      error('need a vector containing the intensities along a line
profile & a search starting point (optional)');
end

dip = gradient(ip);
count = 0;
xs(1) = -1;
ys(1) = -1;
rsign = 0;
sign = 0;

[rxs, rys, rcount, rsign]=peaks(dip, std_dev);
temp=1;
i=1;

while i<=rcount
      if rcount-i <=2 & rcount-i >= 0
            xs(count+1)=rxs(i);
            ys(count+1)=rys(i);
            sign(count+1)=rsign(i);
            count=count +1;
            i=i+1;
      else
            i=i+2;
            if rxs(i)-rxs(temp) > 0.5*pspacing
                  xs(count+1)=rxs(temp);
                  ys(count+1)=rys(temp);
                  sign(count+1)=rsign(temp);
                  xs(count+2)=rxs(i-1);
                  ys(count+2)=rys(i-1);
                  sign(count+2)=rsign(i-1);
                  count = count +2;
                  temp=i;
            else
                  if rsign(temp)*dip(rxs(temp))<dip(rxs(i))
                        temp=i;
                  end
            end
      end
end
pcount=0;
if xs <0 return; end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
function [rxs, rys, rcount, rsign] = peaks(ip, std_dev);%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

if nargin < 1
```

```
      error('need a vector containing the intensities along a line
profile');
end
dip = gradient(ip);
[finish,y] = size(ip);
rcount = 0;
i      = 1;
temp   = i;
rsign = 0;

while i<finish
      if (dip(i)*dip(i+1) <= 0)
            temp = i;
            while ((dip(i)*dip(i+1)) == 0)
                  i = i + 1;
                  if i==finish
                        return;
                  end
            end
            if (i == temp) & i < finish
                  i = i + 1;
            end
            if ((dip(temp)*dip(i)) < 0 )   | (i == finish & temp <
finish)
                  if i-temp ==1
                        if dip(temp) < 0
                              ppos=temp;
                        else
                              ppos=i;
                        end
                  else
                        ppos=round((i+temp)/2);
                  end
                  rxs(rcount+1) = ppos;
                  rys(rcount+1) = ip(ppos);
                  rcount = rcount + 1;
                  if rcount > 0
                        if (dip(temp) > 0) & rcount > 0
                              rsign(rcount) = 1;
                        elseif dip(temp) < 0
                              rsign(rcount) = -1;
                        end
                  end
            end

      else
            i = i+1;
      end
end
if rcount > 0
      if rsign(rcount) == -1
            if ip(length(ip)) > ip(rxs(rcount))
                  rxs(rcount+1)=length(ip);
                  rys(rcount+1)=ip(length(ip));
                  rcount=rcount+1;
                  rsign(rcount)=1;
            end
      end
end
if rcount < 1;
      rxs(1)=-1;
```

```
      rys(1)=-1;
end
```

```
function [Points]=coord(coords, xs, pspacing)

Points=[];
x0=coords(1:2);
x1=coords(3:4);
len=norm(x1-x0)*pspacing;


for r=1:length(xs),
      x=(xs(r)/len)*(x1-x0) + x0;
      %figure(1);
      %plot(x(1,2), x(1,1),'g-o');
      %hold on;
      Points=[Points; x];
end
```

Computer code used for 3-D analysis

```
function V2 = anzload(anzfile,x,y,z,anzprecision,machineformat)
if nargin == 5
    machineformat = 'l';
elseif nargin > 6 | nargin < 5
    s1 = sprintf('need 5 (or an optional 6th) arguments:\n');
    s2 = sprintf('1: the filename of an Analyze image (.img)\n');
    s3 = sprintf('2,3,4: voxel dimensions of the image\n');
    s4 = sprintf('5: datatype used in the image\n');
    s5 = sprintf('6: (optional) format of the data\n');
    s = strcat(s1,s2,s3,s4,s5);
    error(s);
end
fid = fopen(anzfile, 'r', machineformat);
if fid == -1
    s = sprintf('cannot open file \"%s\"\n', anzfile);
    error(s);
end
[VOL,count] = fread(fid, [x*y*z], anzprecision);
fclose(fid);
V2 = reshape(VOL, [x,y,z]);
```

```
function [start, stop] = boundary_pts(lp,c_xs, c_ys, c_count,spacing, grey_range,
c_sign, std_dev, VOX);
if nargin < 1, error('need a line profile vector'); end
dlp = gradient(lp);
[x,y]=size(lp);
magnitudes = zeros(c_count,1);
```

```
start = -1;
stop = -1;
max_peak = -1;
max_mag = [0,0];
if c_count <= 1, return; end
for j=1:c_count
    if c_ys(j) > max_peak
        max_peak = c_ys(j);
    end
end
if max_peak <= 0, return; end
for i=1:c_count
    e1 = c_ys(i) / max_peak ;
    e2 = (x - c_xs(i)) / x;
    tmp = abs(grey_range(1) - lp(c_xs(i)));
    if lp(c_xs(i)) > grey_range(1)
        e3 = (1 - tmp/(grey_range(2)-grey_range(1)));
    else
        e3 = (1 - 4*tmp/(grey_range(2)-grey_range(1)));
    end
    if e3 < 0
        e3 = 0;
    end
    if c_sign(i) > 0
        magnitudes(i) = sqrt(e1^2 + e2^2 + e3^2);
    else
        magnitudes(i) = 0;
    end
end
for i=1:c_count-1
    if i < c_count
        if magnitudes(i)> max_mag(2)
            max_mag = [i,magnitudes(i)];
        end
    end
end
base = x;
for i=1:x
    if lp(i) < 2
        base = i;
        break;
    end
end
if max_mag(1) <= c_count - 2
    e = max_mag(1)+ 2;
elseif max_mag(1) >c_count - 2
    e = max_mag(1);
end
if max_mag(1) <= 2
    b = max_mag(1);
```

```
elseif max_mag(1) > 2
    b = max_mag(1) - 2;
end
if b <= 0
    start = -1;
    stop = -1;
    return;
end
temp = c_ys(b);
start=c_xs(b);
while b < e
    if c_sign(b) > 0 & c_ys(b) > temp & abs(c_xs(b)-c_xs(max_mag(1))) <
max(VOX)/(2*spacing)
        temp = c_ys(b);
        start = c_xs(b);
    end
    b = b + 1;
end
if base > start
    start = -1;
    break;
end
magnitudes = zeros(c_count,1);
for i=1:c_count
    e1 = 0;
    e2 = 0;
    e3 = 0;
    if ((c_xs(i)-start)*spacing < 5.5) & ((c_xs(i)-start)*spacing > 2)
        e1 = c_ys(i) / max_peak;
        e2 = 1-abs((start+(4/spacing) - c_xs(i)) / (start+(4/spacing)));
        if lp(c_xs(i)) < grey_range(2)
            e3 = (1-abs(grey_range(2)-lp(c_xs(i)))/(grey_range(2)-grey_range(1)));
        else
            e3 = (1-4*abs(grey_range(2)-lp(c_xs(i)))/(grey_range(2)-
grey_range(1)));
        end
    end
    if e1 <= 0 | e2 <= 0 | e3 <= 0
        e1 = 0;
        e2 = 0;
        e3 = 0;
    end
    if c_sign(i) > 0
        if c_xs(i) ~= start
            magnitudes(i) = sqrt(e1^2 + e2^2 + e3^2);
        else
            magnitudes(i) = 0;
        end
    else
        magnitudes(i) = 0;
```

```
    end
end
max_mag = [0,0];
for i=1:c_count
    if magnitudes(i)> max_mag(2)
        max_mag = [i,magnitudes(i)];
    end
end
if max_mag(1) ~= 0
    stop = c_xs(max_mag(1));
else
    stop = -1;
end
if base > start, stop = -1;
break;
end
```

```
function [R,G,B] = colour_picker(drops, max_drop, widths, num_lam, type);
if nargin ~= 5
    error('need drop, max drop, width, number of lamina and type of colouring');
end
if exist('DEFINE_SCHEME') ~= 1
    colour_scheme
end
switch type
    case { NUM_LAM }
        if num_lam <= 0
            R = 255;
            G = 255;
            B = 255;
        elseif num_lam == 1
            R = 0;
            G = 0;
            B = 255;
        elseif num_lam == 2
            R = 0;
            G = 255;
            B = 0;
        else
            R = 255;
            G = 0;
            B = 0;
        end
    case { FIRST_DROP }
        if drops(1) > 0
            h = 0; s = 1; v = 255;
            s = drops(1)/max_drop;
            [R,G,B] = hsv2rgb([h,s,v]);
```

```
            round(R); round(G); round(B);
        elseif drops(1) < 0
            R = 0;
            G = 0;
            B = 255;
        else
            R = 255;
            G = 255;
            B = 255;
        end
    otherwise
        if num_lam > 1
            maxd = max(drops);
            for i=1:num_lam
                if drops(i) == maxd, break; end
            end
            h = 0; s = 1; v = 255;
            s = drops(i) / 10;
            [R,G,B] = hsv2rgb([h,s,v]);
            R = round(R); G = round(G); B = round(B);
        elseif num_lam == 1
            h = 0; s = 1; v = 255;
            s = drops(1) / 10;
            [R,G,B] = hsv2rgb([h,s,v]);
            R = round(R); G = round(G); B = round(B);
        elseif num_lam == 0
            R = 255; G = 255; B = 255;
        else
            R = 0; G = 0; B = 255;
        end
end
```

```
global DEFINE_SCHEME;
global NUM_LAM FIRST_DROP MAX_DROP FIRST_WIDTH MAX_WIDTH;
NUM_LAM    = 1;
FIRST_DROP  = 2;
MAX_DROP    = 3;
FIRST_WIDTH = 4;
MAX_WIDTH   = 5;
```

```
function [xs, ys, count, sign] = concavity(lp, std_dev);
if nargin < 2 | nargin > 3
    error('need a vector containing the intensities along a line profile & a search
starting point (optional)');
end
dlp = gradient(lp);
[x,y] = size(lp);
[xs, ys, count, sign] = peaks(dlp, 1, y, std_dev);
```

```
function [drops, widths, num_lam] = lpa(lp, spacing, num, std_dev, pl, tissue_type,
grey_range, VOX);
if nargin ~= 8
    error('wrong number of input arguments');
end
OK = 1;
BAD = 0;
drops = 0;
widths = [0,0,0];
tag = OK;
app_s = '';
if nargin == 2, pl = 0; end
[x,y] = size(lp);
xmin = 0;
xmax = lp(x,1);
ymin = -5;
ymax = grey_range(2)*2;
[c_xs, c_ys, c_count, c_sign] = concavity(lp(:,2)', std_dev*spacing^4);
[start,stop] = boundary_pts(lp(:,2),c_xs, c_ys, c_count, spacing, grey_range,c_sign,
std_dev*spacing, VOX);
if start < 0 | stop < 0
    tag = BAD;
    app_s = strcat(app_s,' [cannot find boundaries]');
    num_lam = -1;
end
[s_xs, s_ys, s_count, s_sign] = stationary(lp(:,2)', start, stop, tissue_type,std_dev,
spacing);
num_lam = round(s_count/2);
if ((s_count == 0) | (s_xs(s_count) < start)) & (tag ~= BAD)
    app_s = strcat(app_s,' [no stationary pts]');
end
ind = 1;
if tag == BAD
    drops(1) = -1;
    num_lam = -1;
elseif s_count > 1 & start >= 0
    for j=1:(s_count-1)
        tmp = s_ys(j) - s_ys(j+1);
        if tmp > 0
            if std_dev ~= 0
                drops(ind) = tmp / std_dev;
            else
                drops(ind) = tmp;
            end

            for k=1:(c_count-1)
                if c_xs(k) < s_xs(j) & c_xs(k+1) > s_xs(j)
                    ca = c_xs(k)-s_xs(j);
```

```
                    cb = (c_xs(k)-s_xs(j))/(s_xs(j+1) - s_xs(j));
                    cc = (s_xs(j+1) - s_xs(j));
                    widths(ind,:) = [ca,cb,cc];
                else
                    ca = (s_xs(j+1) - s_xs(j))/2;
                    cb = 0;
                    cc = (s_xs(j+1) - s_xs(j));
                    widths(ind,:) = [ca,cb,0];
                end
                widths(ind,:) = widths(ind,:).*spacing;
            end
            ind = ind + 1;
        end
    end
end
if pl <= 0, return; end
scale1st_diff = 20;
scale2nd_diff = 100;
if tag ~= BAD
    dlp = gradient(lp(:,2));
    ddlp = gradient(dlp);
    plot(lp(:,1), dlp.*scale1st_diff,'r');
    hold on;
    plot(lp(:,1), ddlp.*scale2nd_diff,'g');
end
plot(lp(:,1),lp(:,2),'b');
hold on;
if s_count > 0, plot(lp(s_xs(1),1), s_ys(1),'r+'); end
if s_count > 1, plot(lp(s_xs(2),1), s_ys(2),'g+'); end
if start > 0 & stop > 0
    plot(start*spacing, lp(start,2), 'k*');
    plot(stop*spacing, lp(stop,2), 'k*');
end
grid on;
axis([xmin, xmax, ymin, ymax]);
if tag == OK
    foo = strcat('line profile (max lamination drop = ', num2str(drops(1)), ')' ,app_s);
    title(foo);
else
    foo = strcat('line profile (DUD)',app_s);
    title(foo);
end
xlabel('distance (mm)');
ylabel('intensity');
hold off;
pause;
```

```
function [newx, newy, newz] = normalize(oldx, oldy, oldz);
if nargin ~= 3, error('normalize works on 3D points/vectors only'); end
```

```
ssum = sqrt(abs(oldx)^2 + abs(oldy)^2 + abs(oldz)^2);
if ssum == 0,
    warning('cannot find normal, possible degenerate triangle.');

    newx = 0;
    newy = 0;
    newz = 0;
else
    newx = oldx / ssum;
    newy = oldy / ssum;
    newz = oldz / ssum;
end
```

```
function [rxs, rys, rcount, rsign] = peaks(lp, start, finish, std_dev);
if nargin < 1
    error('need a vector containing the intensities along a line profile');
end
dlp = gradient(lp);
[x,y] = size(lp);
count = 0;
rcount = 0;
i    = start;
temp  = i;
sign = 0;
rsign = 0;
while i<finish
    if (dlp(i)*dlp(i+1) <= 0)
        temp = i;
        while (abs(dlp(i)*dlp(i+1)) <= std_dev)& (i<finish-1)
            i = i + 1;
        end
        if i == temp & i < finish
            i = i + 1;
        end
        if ((dlp(temp)*dlp(i)) < 0 & dlp(temp) ~= dlp(i))  | i == finish-1
            rxs(rcount+1) = round((i + temp)/2);
            rys(rcount+1) = lp(rxs(rcount+1));
            rcount = rcount + 1;
            if rcount > 0
                if dlp(temp) > 0 & rcount > 0
                    rsign(rcount) = 1;
                elseif dlp(temp) < 0
                    rsign(rcount) = -1;
                end
            end
        end

    else
        i = i+1;
```

```
    end
end
if rcount <= 1;
    rxs(1)=-1;
    rys(1)=-1;
end
```

```
% quickrun.m

clear all;
close;
format long
tic
if exist('DEFINE_SCHEME') ~= 1
    colour_scheme
end
ot = MAX_DROP;
nl = sprintf('\n');
disp('Please enter the name of a mesh file.');
fin = deblank(input('mesh file: ','s'));
[path, base, ext, ver] = fileparts(fin);
s = sprintf('output will be in file \"%s.off\"\n', base);
disp(s);
fout = strcat(base,'.out');
fid = fopen(fout,'w');
disp('Depth (mm) the line profile should reach into the cortex?');
depth = input('depth: ');
disp(nl);
disp('Number of sample points to interpolate?');
samples = input('samples: ');
disp(nl);
disp('Standard deviation of background noise?');
disp('(Enter the value 1, to ignore signifigance rel. to background noise.)');
std_dev = input('std. deviation: ');
disp(nl);
disp('Please enter the lower limit of the grey matter intensity range?');
low = input('intensity value: ');
disp(nl);
disp('Please enter the upper limit of the grey matter intensity range?');
high = input('intensity value: ');
grey_range = [low,high];
disp(nl);
disp('What type of tissue analysis do you wish to perform?');
disp('choices: ');
disp('"g"   : grey matter analysis');
disp('"w"   : white matter analysis');
tissue_type = input('tissue type: ','s');
disp(nl);
disp('What kind of analysis do you want to perform?');
```

```
disp('choices: ');
s=sprintf('"%d"      : colour coded for number of laminations',NUM_LAM);
disp(s);
s=sprintf('"%d"      : colour coded for first intensity drop',FIRST_DROP);
disp(s);
disp('"0"      : colour coded for maximum intensity drop');
ot = input('analysis type: ');
hfile = strcat(path, '/', base, '.img');
switch lower(ext)
    case {'.wfr'}
        [num_triangles, num_points, triangles, points] = read_emse(fin,hfile);
    otherwise
        s = sprintf('unknown mesh type \"%s\"\n',ext);
        error(s);
end
[DIM VOX SCALE TYPE OFFSET ORIGIN DESCRIP] = spm_hread(hfile);
fprintf(fid,'# %d\n', num_triangles);
p1 = zeros(1,3);
p2 = zeros(1,3);
p3 = zeros(1,3);
j=1;
for i=1:1:num_triangles
    p1 = points(triangles(i,1)+1,:);
    p2 = points(triangles(i,2)+1,:);
    p3 = points(triangles(i,3)+1,:);

    p1(1) = p1(1) * VOX(1);
    p2(1) = p2(1) * VOX(1);
    p3(1) = p3(1) * VOX(1);
    p1(2) = p1(2) * VOX(2);
    p2(2) = p2(2) * VOX(2);
    p3(2) = p3(2) * VOX(2);
    p1(3) = p1(3) * VOX(3);
    p2(3) = p2(3) * VOX(3);
    p3(3) = p3(3) * VOX(3);

    [cx, cy, cz] = tricenter(p1, p2, p3);
    [nx, ny, nz] = trinormal(p1, p2, p3);
    [nx, ny, nz] = normalize(nx,ny,nz);
    cx = cx / VOX(1); nx = nx / VOX(1);
    cy = cy / VOX(2); ny = ny / VOX(2);
    cz = cz / VOX(3); nz = nz / VOX(3);
    if [nx,ny,nz] == [0,0,0];
        disp('triangle number:');
        disp(i);
    else
        fprintf(fid,'s %f %f %f  d %f %f %f\n',cx,cy,cz, nx,ny,nz);
        j = j + 1;
    end
end
```

```
fclose(fid);
snc(base, depth, samples, std_dev, ot, 0, tissue_type, grey_range);
toc
```

```
function [num_tri, num_points, triangles, points] = read_emse(fname, hname);
if nargin ~= 2
    error('need meshfile and header file');
end
fid = fopen(fname, 'rt');
if fid == -1
    s = sprintf('cannot open file \"%s\"\n', fname);
    error(s);
end
[path, base, ext, ver] = fileparts(fname);
[DIM VOX SCALE TYPE OFFSET ORIGIN DESCRIP] = spm_hread(hname);
output_file = strcat(base, '.off');
tri_file    = strcat(base, '.tri');
fout = fopen(output_file,'w');
for i=1:1:3, fgetl(fid); end
str = sprintf('grep -c v %s', fname);
[s,w] = unix(str);
num_points = str2num(w)
points = zeros(num_points, 3);
str = sprintf('grep t %s | sed \"s/t/3/g\" > %s', fname, tri_file);
unix(str);
str = sprintf('grep -c t %s', fname);
[s,w] = unix(str);
num_tri = str2num(w)
triangles = zeros(num_tri, 3);
fprintf(fout,'OFF\n%d %d %d\n', num_points, num_tri, 0);
i=1;
p_ind=1; t_ind=1;
line = fgetl(fid);
while line ~= -1,
    P = sscanf(line, 'v %f %f %f')';
    if ~isempty(P)
        a = P(1); b = P(2); c = P(3);
        points(p_ind,:)=[a,b,c];
        p_ind = p_ind + 1;
        fprintf(fout,'%f %f %f\n', a*VOX(1), b*VOX(2), c*VOX(3));
    else
        T = sscanf(line, 't %i %i %i')';
        if ~isempty(T)
            triangles(t_ind,:)=T;
            t_ind = t_ind + 1;
        end
    end
```

```
    line = fgetl(fid);
end
fclose(fid);
```

```
function snc(base, depth, N, std_dev, ot, pl, tissue_type, grey_range);
if nargin < 7 | nargin > 8;
    error('wrong number of arguments');
end
if exist('DEFINE_SCHEME') ~= 1
    colour_scheme
end
output_file = strcat(base, '.off');
int_file   = strcat(base, '.out');
tri_file   = strcat(base, '.tri');
img_file   = strcat(base, '.img');
[DIM VOX SCALE TYPE OFFSET ORIGIN DESCRIP] = spm_hread(img_file);
vol_data = anzload(img_file, DIM(1), DIM(2), DIM(3), 'uint8', 'l');
fout = fopen(output_file,'a');
fid = fopen(int_file, 'r');
ft = fopen(tri_file,'r');
line = fgetl(fid);
num_triangles = sscanf(line,'# %d');
k = depth/N;
for i=1:num_triangles
    line = fgetl(fid);
    tri = fgetl(ft);
    A = sscanf(line,'s %f %f %f d %f %f %f');
    x0 = A(1); y0 = A(2); z0 = A(3);
    nx = A(4); ny = A(5); nz = A(6);
    dx = k*nx; dy = k*ny; dz = k*nz;
    x = (x0 + (-N/2:N-1)*dx);
    y = (y0 + (-N/2:N-1)*dy);
    z = (z0 + (-N/2:N-1)*dz);
    scalev = (1:3/2*N) * k;
    lp = interp3(vol_data,x,y,z,'*linear');
    newlp = [ scalev',lp'];
    [drops, widths, num_lam] = lpa(newlp, k, i, std_dev, pl, tissue_type, grey_range,
VOX);
    max_drop = grey_range(2) - grey_range(1);
    [R, G, B] = colour_picker(drops, max_drop, widths, num_lam, ot);
    fprintf(fout,'%s %d %d %d\n', tri, R, G, B);
end
fclose(fid);
delete(tri_file);
```

```
function [xs, ys, count, sign] = stationary(lp, agbound, gwbound, tissue_type, std_dev,
spacing);
```

```
if nargin < 5 | nargin > 6
    error('need a vector containing the intensities along a line profile & a search
starting point (optional)');
end
[x,y] = size(lp);
count = 0;
xs(1) = -1;
ys(1) = -1;
rsign = 0;
sign = 0;
if tissue_type == 'g'
    start = agbound;
    stop = gwbound;
elseif tissue_type == 'w'
    start = gwbound;
    stop = y;
else
    error('unknown tissue type');
end
if stop < 0, return; end
[rxs, rys, rcount, rsign]=peaks(lp, start, stop, std_dev*spacing^2);
if rcount > 2
    if rys(rcount) > rys(rcount -1)
        rcount=rcount - 1;
    end
end
for i=1:2:rcount-1
    if lp(rxs(i))-lp(rxs(i+1)) > 2 * std_dev
        xs(count+1)=rxs(i);
        xs(count+2)=rxs(i+1);
        ys(count+1)=rys(i);
        ys(count+2)=rys(i+1);
        count=count+2;
        sign(count+1)=rsign(i);
        sign(count+2)=rsign(i+1);
    end
end
```

```
% testrun.m

clear all;
close;
format long
tic
if exist('DEFINE_SCHEMES') ~= 1
    colour_scheme
end
ot = MAX_DROP;
nl = sprintf('\n');
```

```
disp('Please enter the name of a mesh file.');
fin = which(deblank(input('mesh file: ','s')));
[path, base, ext, ver] = fileparts(fin);
s = sprintf('output will be in file \"%s.off\"\n', base);
disp(s);
fout = strcat(base,'.out');
fid = fopen(fout,'w');
disp('Depth (mm) the line profile should reach into the cortex?');
depth = input('depth: ');
disp(nl);
disp('Number of sample points to interpolate?');
samples = input('samples: ');
disp(nl);
disp('Standard deviation of background noise?');
disp('(Enter the value 1, to ignore signifigance rel. to background noise.)');
std_dev = input('std. deviation: ');
disp(nl);
disp('Please enter the lower limit of the grey matter intensity range?');
low = input('intensity value: ');
disp(nl);
disp('Please enter the upper limit of the grey matter intensity range?');
high = input('intensity value: ');
disp(nl);
disp('What type of tissue analysis do you wish to perform?');
disp('choices: ');
disp('"g"   : grey matter analysis');
disp('"w"   : white matter analysis');
tissue_type = input('tissue type: ','s');
disp(nl);
grey_range = [low,high];
hfile = strcat(path, '/', base, '.img');
switch lower(ext)
      case {'.wfr'}
            [num_triangles, num_points, triangles, points] = read_emse(fin,hfile);
      otherwise
            s = sprintf('unknown mesh type \"%s\"\n',ext);
            error(s);
end
[DIM VOX SCALE TYPE OFFSET ORIGIN DESCRIP] = spm_hread(hfile);
fprintf(fid,'# %d\n', num_triangles);
p1 = zeros(1,3);
p2 = zeros(1,3);
p3 = zeros(1,3);
j=1;
for i=1:1:num_triangles
      p1 = points(triangles(i,1)+1,:);
      p2 = points(triangles(i,2)+1,:);
      p3 = points(triangles(i,3)+1,:);
      p1(1) = p1(1) * VOX(1);
      p2(1) = p2(1) * VOX(1);
```

```
    p3(1) = p3(1) * VOX(1);
    p1(2) = p1(2) * VOX(2);
    p2(2) = p2(2) * VOX(2);
    p3(2) = p3(2) * VOX(2);
    p1(3) = p1(3) * VOX(3);
    p2(3) = p2(3) * VOX(3);
    p3(3) = p3(3) * VOX(3);
    [cx, cy, cz] = tricenter(p1, p2, p3);
    [nx, ny, nz] = trinormal(p1, p2, p3);
    [nx, ny, nz] = normalize(nx,ny,nz);
    cx = cx / VOX(1); nx = nx / VOX(1);
    cy = cy / VOX(2); ny = ny / VOX(2);
    cz = cz / VOX(3); nz = nz / VOX(3);
    if [nx,ny,nz] == [0,0,0];
        disp('triangle number:');
        disp(i);
    else
        fprintf(fid,'s %f %f %f  d %f %f %f\n',cx,cy,cz, nx,ny,nz);
        j = j + 1;
    end
end
fclose(fid);
snc(base, depth, samples, std_dev, ot, 1, tissue_type, grey_range);
toc
```

```
function [x,y,z] = tricenter(p1, p2, p3);
x = (p1(1)+p2(1)+p3(1))/3;
y = (p1(2)+p2(2)+p3(2))/3;
z = (p1(3)+p2(3)+p3(3))/3;
```

```
function [cx,cy,cz] = trinormal(p1, p2, p3);
if nargin ~= 3, error('need three 3D points as input'); end
ax = p1(1) - p2(1);
ay = p1(2) - p2(2);
az = p1(3) - p2(3);
bx = p3(1) - p2(1);
by = p3(2) - p2(2);
bz = p3(3) - p2(3);
cx = ay*bz - az*by;
cy = az*bx - ax*bz;
cz = ax*by - ay*bx;
```

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. A method of mapping a property of a three dimensional object, said method comprising the steps of
- mapping the property for at least a portion of the object,
- providing a line or region which defines intersections with part of said mapped portion, and
- displaying the property for the intersections.

2. A method of mapping a property of a three dimensional object, said method comprising the steps of
- mapping the property for a plurality of slices within the object,
- providing a line or region which defines intersections with part of at least some of the plurality of slices, and
- displaying the property for the intersections.

3. A method as defined in claims 1 or 2 wherein the property relates to nuclear magnetic resonances.

4. A method as defined in claim 2 wherein mapping of the property for the plurality of slices results in a plurality of nuclear magnetic resonance images.

5. A method as defined in any one of the preceding claims wherein the line or region is a line.

6. A method as defined in claim 5 wherein the line is a plurality of lines.

7. A method as defined in claim 6 wherein each of the lines is substantially perpendicular to the surface of the three-dimensional object.

8. A method as defined in any one of the preceding claims which further comprises the step of approximating the shape of the object by a three dimensional lattice of two-dimensional forms created by a first computer routine.

9. A method as defined in claim 8 wherein the lattice is a mesh.

10. A method as defined in any one of claims 8 to 9 wherein each of the lines is a line substantially perpendicular to one of the two-dimensional forms.

11. A method as defined in claim 2 wherein the slices are selected using a second computer routine.

12. A method as defined in any one of claims 3 to 11 wherein a third computer routine maps nuclear magnetic resonance values for the intersections by displaying an array of values.

13. A method as defined in claim 12 which further comprises the step of analysing the array of values using a fourth computer routine.

14. A method as defined in claim 13 wherein the fourth computer routine analyses numerical properties.

15. A method as defined in any one of claims 12 to 14 wherein the step of analysing the array of values comprises differentiation.

16. A method as defined in claim 15 wherein analysing the array of values comprises determining the positions of the maxima, minima and/or zero-points of the array of values and the derivatives of the array of values.

17. A map produced by a method of mapping a property of a three dimensional object, said method comprising the steps of
- mapping the property for at least a portion of the object,
- providing a line or region which defines intersections with part of said mapped portion, and
- displaying the property for the intersections.

18. A map produced by a method of mapping a property of a three dimensional object, said method comprising the steps of
- mapping the property for a plurality of slices within the object,
- providing a line or region which defines intersections with part of at least some of the plurality of slices, and
- displaying the property for the intersections.

19. A method of mapping a property of an object, said method comprising the steps of
- mapping the property for a slice within the object,
- providing a line within the slice, and
- displaying the property for the line.

20. A method as defined in claim 19 wherein the property wherein relates to nuclear magnetic resonances.

21. A method as defined in any one of claims 19 to 20 wherein mapping of the property for the slice results in a nuclear magnetic resonance image.

22. A method as defined in any one of claims 19 to 21, wherein the line is a plurality of lines.

23. A method as defined in any one of claims 19 to 22 wherein a fifth computer routine maps nuclear magnetic resonance values for the intersections by displaying an array of values.

24. A method as defined in claim 23 which further comprises the step of analysing the array of values using a sixth computer routine.

25. A method as defined in claim 24 wherein the fifth computer routine analyses numerical properties.

26. A method as defined in any one of claims 23 to 25 wherein the step of analysing the array of values comprises differentiation.

27. A method as defined in claim 26 wherein the step of analysing the array of values comprises determining the positions of the maxima, minima and/or zero-points of the array of values and the derivatives of the array of values.

28. A map produced by a method of mapping a property of an object, said method comprising the steps of

- mapping the property for a slice within the object,
- providing a line within the slice, and
- displaying the property for the line.

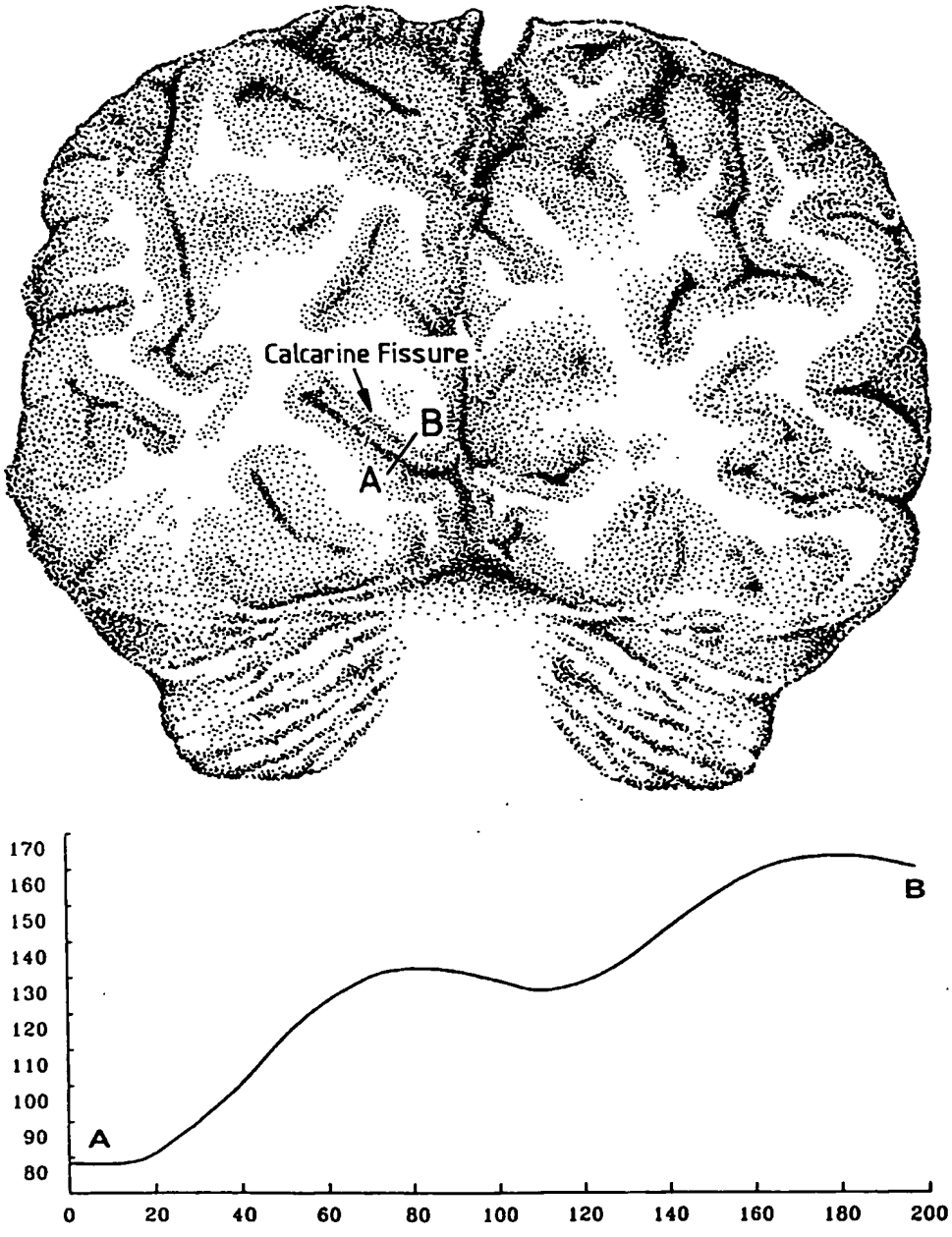
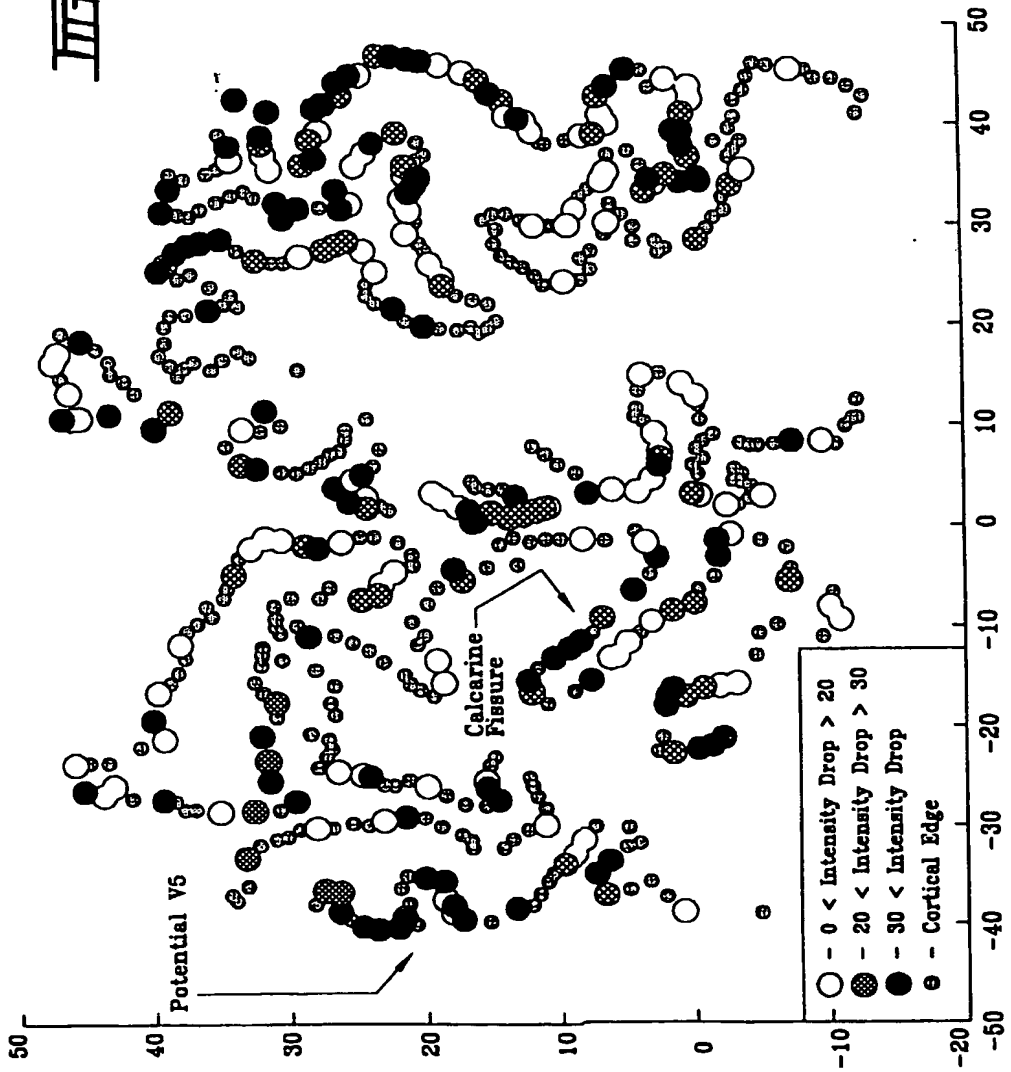

Calcarine Fissure
B
A
170
160
150
140
130
120
110
100
90
80
A
B
0
20
40
60
80
100
120
140
160
180
200

Fig. 1a

FIG. 1b.


Potential V5
Calcarine Fissure
- 0 < Intensity Drop > 20
- 20 < Intensity Drop > 30
- 30 < Intensity Drop
- Cortical Edge
50
40
30
20
10
0
-10
-20
-50
-40
-30
-20
-10
0
10
20
30
40
50

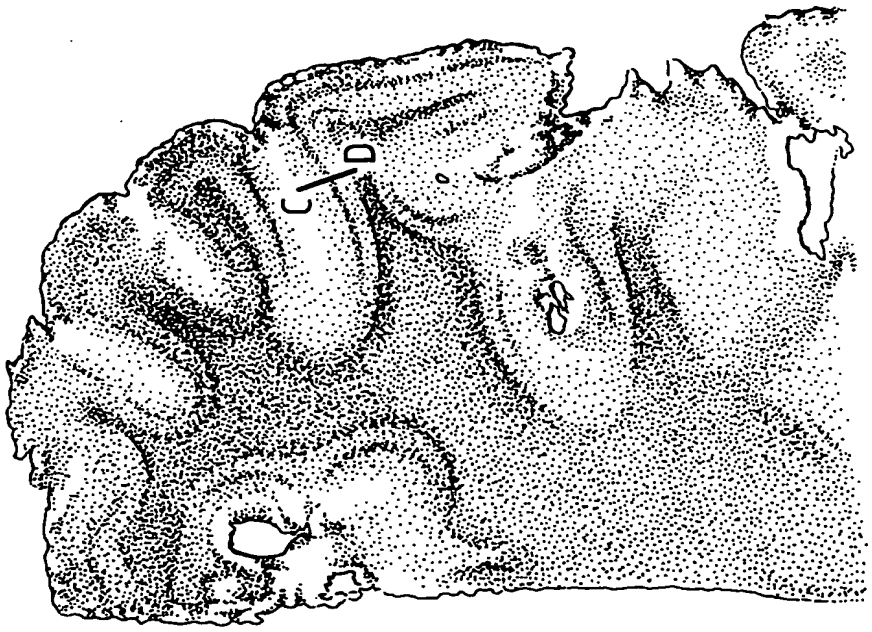


Fig. 2(ii).

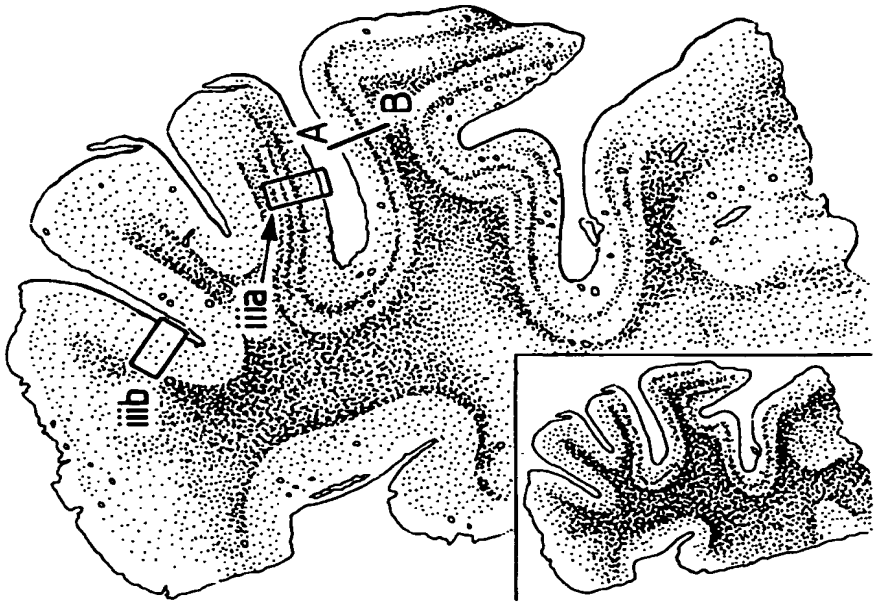


Fig. 2(i).

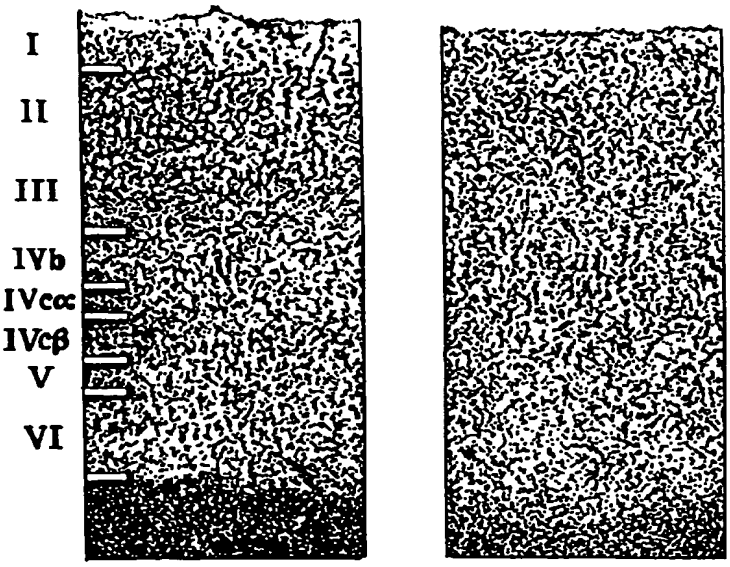


FIG. 2(iiia) FIG. 2(iiib)

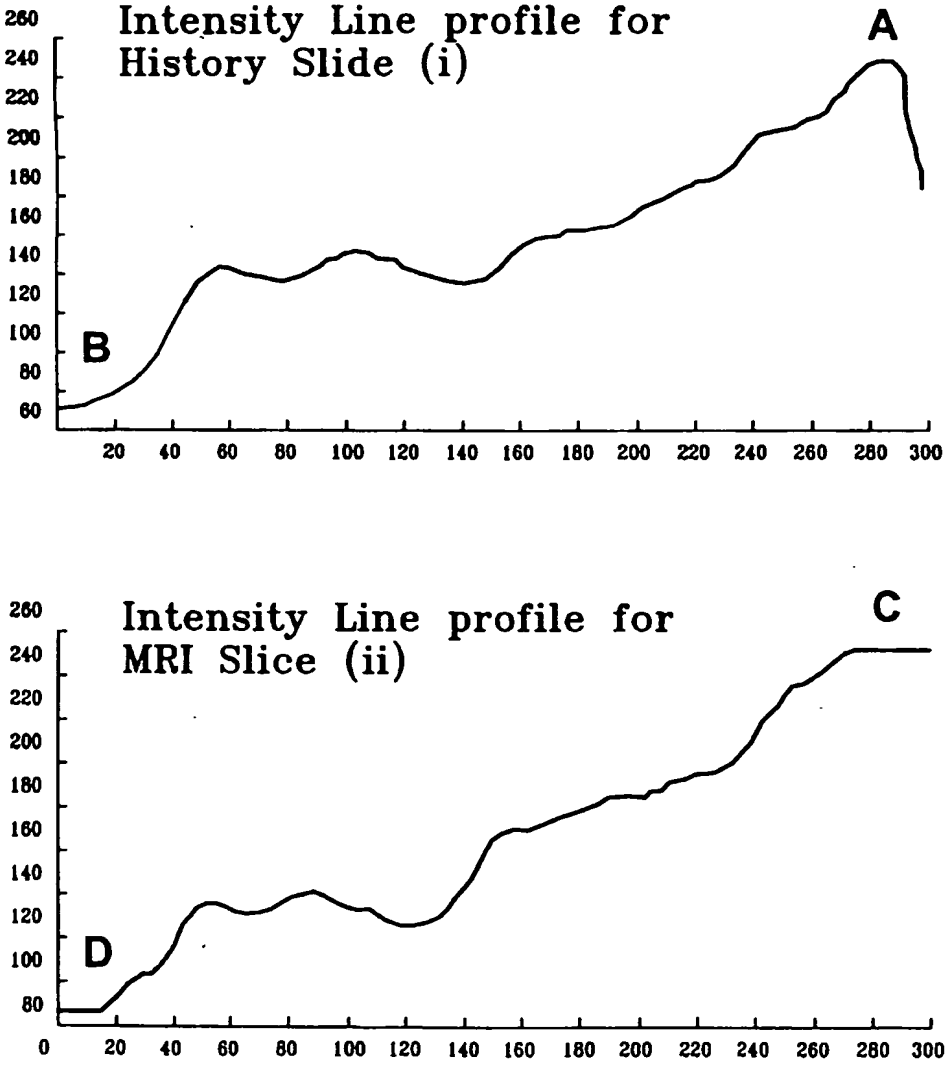

Intensity Line profile for History Slide (i)
A
B
260
240
220
200
180
160
140
120
100
80
60
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
Intensity Line profile for MRI Slice (ii)
C
D
260
240
220
200
180
160
140
120
100
80
0
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300

Fig. 2(iv).

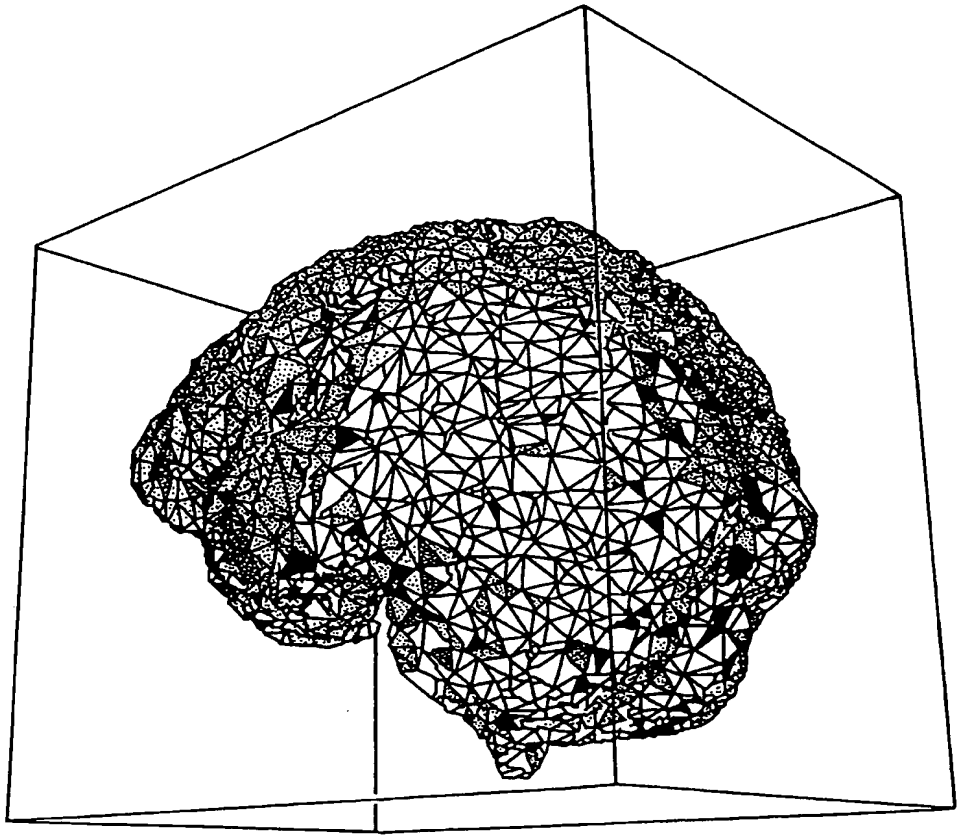


FIG. 3.

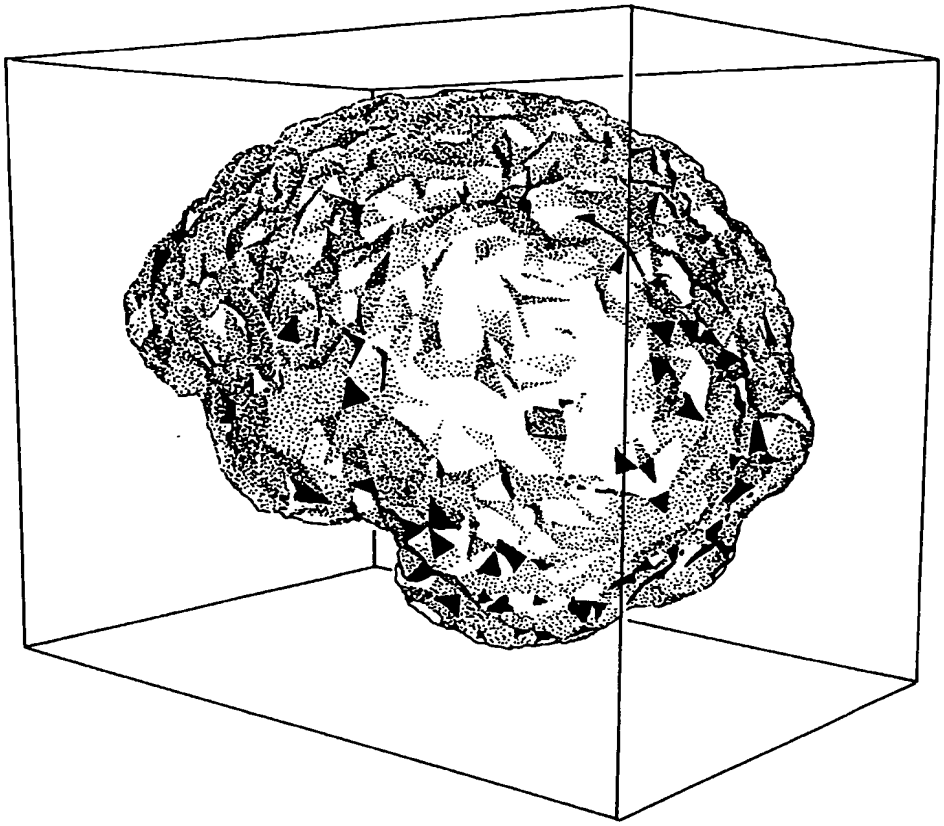


Fig. 4.

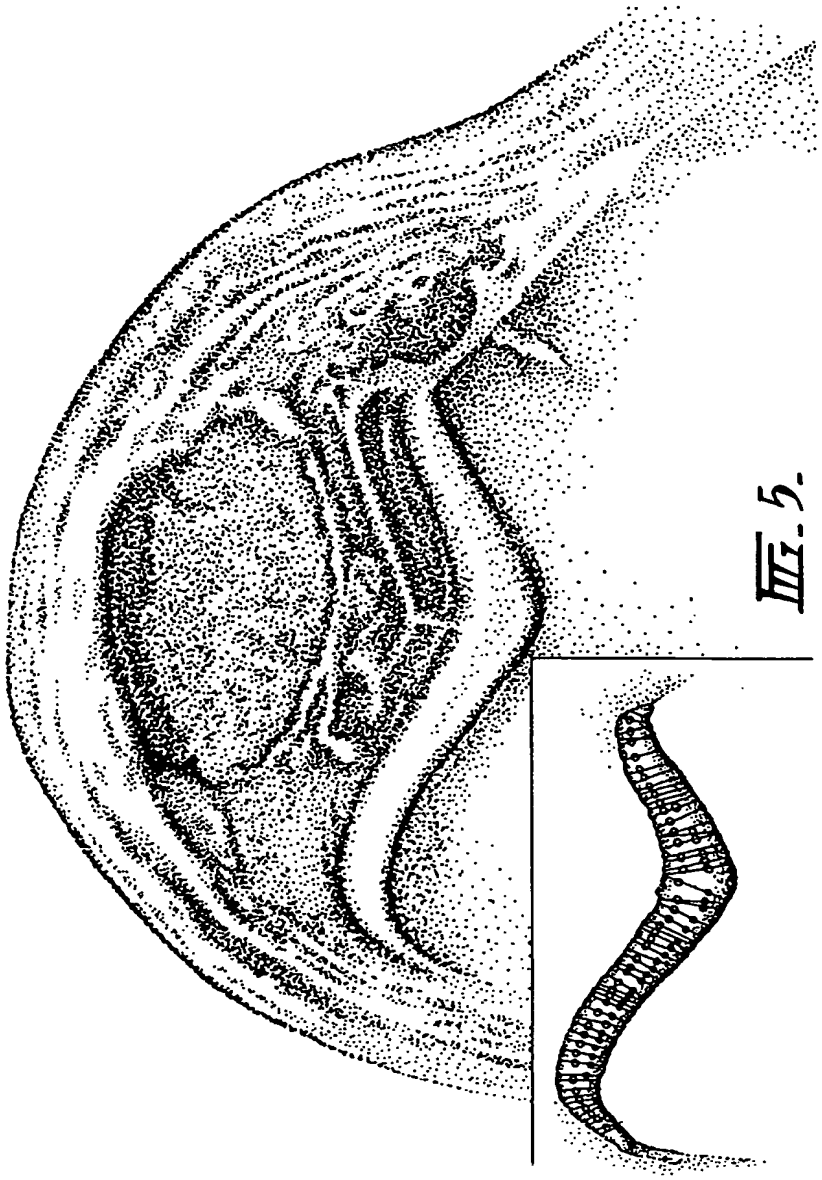


Fig. 5.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU02/00731

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: A61B 5/055

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
DWPI Keywords (map, line, intersect, propert, 3D) and like terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 95/20354 A1 (BOARD OF TRUSTEES OF LELAND STANFORD, JR. UNIVERSITY) 3 August 1995 Page 5, line 16 to page 11, line 30 Figs. 1-8	1, 2, 4, 17, 18
X	Derwent Abstract Accession No. 91-022356/03 Class S02, WO 90/16037 A (FUJITSU LTD) 27 December 1990	1, 17
A	EP 521689 B1 (GENERAL ELECTRIC COMPANY) 15 September 1999 Column 4, lines 24-41, Fig. 8	

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 July 2002	9 AUG 2002
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **VINCE BAGUSAUSKAS** Telephone No : (02) 6283 2110

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU02/00731

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4961425 A (KENNEDY ET AL) 9 October 1990 column 2, lines 56-59 column 7, lines 29-56 Figs. 2, 3	
A	US 5262945 A (DECARLI ET AL) 16 November 1993 column 2, lines 8-55 Fig. 3B	
A	Derwent Abstract Accession No. 2002-007824/01 Class P31;S05, JP 2001-291091 A (MITSUBSHI ELECTRIC CORP) 19 October 2001	

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/AU02/00731

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
WO	95/20354	EP	741543	US	5458125		
WO	90/16037	AU	58352/90	EP	431191	US	5446798
EP	521689	JP	5220136	US	5383119		
US	4961425	NONE					
US	5262945	NONE					
JP	2001-291091	NONE					

END OF ANNEX